Adaptive Filtering Using Lyapunov Theory And Artificial Intelligent Techniques

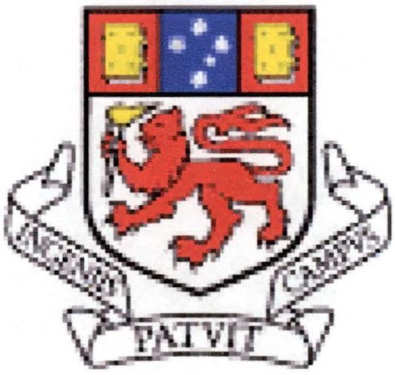
By

Seng Kah Phooi

School of Engineering, University of Tasmania, Australia

Submitted in fulfilment of the requirements

for the degree of Doctor of Philosophy

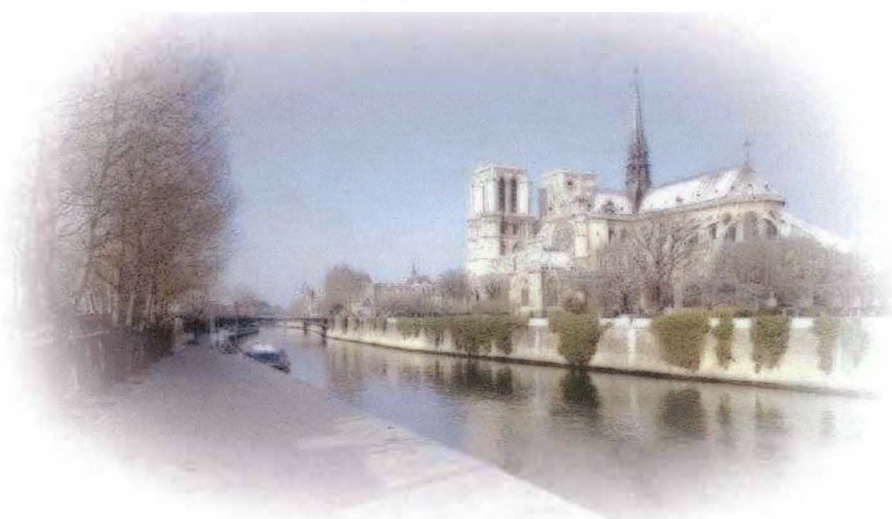


November 2001

Statement of Originality

I hereby declare that the work contained in this PhD thesis is original, to the best of my knowledge and belief, except as acknowledged in the text. The material in this thesis has not been submitted or accepted for the award of any degree or diploma at any tertiary institution. I also declare that the intellectual content of this thesis is the product of my own work, even though I may have received assistance from others on style, presentation and language expression.

Seng Kah Phooi
School of Engineering
University of Tasmania, Australia



Authority of Access

This thesis may be made available for loan. Copying of any part of this thesis is prohibited for two years from the date this statement was signed; after that time limited copying is permitted in accordance with the Copyright Act 1968.

Seng Kah Phooi
School of Engineering
University of Tasmania, Australia

Abstract

Adaptive filtering has gained popularity in numerous applications to help cope with time-variations of system parameters, and to compensate for the lack of *a priori* knowledge of statistical properties of the input data. Therefore, it is an area of research that has important implications for many problems in signal processing, control and estimation, communication and others. Over the last several decades, a wide range of filter structures and algorithms has been developed. *Finite Impulsive Response* (FIR) and *Infinite Impulse Response* (IIR) transversal filters are two well-established linear models for adaptive filtering. However, there are several circumstances that the performances of these filters are unsatisfactory. Nonlinear polynomial filtering had been first considered by some researchers. More recently, artificial intelligent techniques such as neural networks and fuzzy logic have undergone rapid development and become recognized as powerful nonlinear approximation methods. Hence various nonlinear adaptive filtering techniques using multi-layer perceptron (MLP), radial basis function (RBF) and fuzzy logic have been developed.

Adaptive filtering algorithm is another important topic for adaptive filtering. There are two well-studied algorithms for adaptive filtering: *recursive least squares* (RLS) and *least mean square* (LMS) algorithms. LMS algorithm attempts to minimize the mean square of the error signal by employing a stochastic gradient technique. It is strongly dependent on the input signal spectral characteristics and its convergence depends on the eigen-value spread of the autocorrelation matrix. In contrast, several advantages of RLS over LMS in terms of tracking behavior and fast convergence are well known. It is independent of input spectral characteristics but it is of high computational complexity. Furthermore, it exhibits unstable performance. Methods of avoiding instability have been proposed in the literature but the stability problems of adaptive filters have not been solved if there are some bounded input disturbances.

This thesis has provided a fundamental breakthrough in understanding of the Lyapunov stability-based adaptive filtering mechanism, yielding further conditions and solutions for a number of nonlinear filtering problems using Lyapunov stability theory. The first issue to be addressed is the mathematical foundation of Lyapunov stability theory for adaptive filtering systems. Linear models such as FIR and IIR transversal filters using Lyapunov stability theory are developed and analyzed. A new insight is given into the stabilization problem of the adaptive filtering algorithm. The developed Lyapunov stability-based adaptive filtering can provide stability and high tracking precision for adaptive filtering systems. It can overcome the low tracking precision and instability problems of conventional adaptive filtering systems. The designs of those adaptive filters are independent of stochastic properties of signals. The analysis and design of the Lyapunov sense adaptive filters are significantly simplified compared to existing conventional filtering algorithms. The successful outcome of the thesis will in no doubt make significant contributions to and impacts on research in the field of intelligent signal processing and communications systems.

Further investigations presented in this thesis include the theory and design of RBF neural network-based nonlinear adaptive filters with Lyapunov stability, fuzzy adaptive filters with Lyapunov sense fuzzy rules, neural adaptive filters with the back-propagation learning rules in Lyapunov sense with guaranteed stability, polynomial adaptive filters with Lyapunov stability and parallel signal processing using Lyapunov theory. These new adaptive filtering schemes have been implemented to different applications. Simulation examples have been performed to investigate various performances such as tracking precision, stability, and robustness of the developed schemes.

In summary, this thesis has provided an advanced understanding of the Lyapunov stability-based adaptive filtering mechanism. Several adaptive filtering schemes using artificial intelligent techniques and Lyapunov theory have been developed.

Acknowledgements

I wish to acknowledge my thanks to people who have helped me to complete this thesis. Firstly, I would like to express my deepest gratitude to my supervisor Dr Man Zhihong, School of Engineering, University of Tasmania, for his valuable guidance, encouragement and support. I would like to express my appreciation of the time, advice, patience, understanding, willingness and enthusiasm which he has provided. He has helped me greatly throughout my work and the development of the thesis. It has been truly a pleasure to work under the supervision of Dr. Zhihong Man.

Secondly, I sincerely thank Dr Richard Langman for his kindness and advice during my undergraduate and postgraduate studies. I would also like to express my appreciation to Professor Frank Bullen, Head of the School of Engineering, for his support in providing financial assistance in my attending conferences. Thanks are also given to Ms Judy Bonsey who has always been very willing to assist. Many thanks are also extended to other academic and supporting staffs in the School of Engineering.

I would like to express my deep gratitude to my parents, who have made great sacrifices for their children. I would also like to express my thoughts of love and appreciation to my father, who has helped and advised me through many difficulties in my life. My thanks are also dedicated to all my brother and sisters for their care of my parents during the years of my study in Australia. Thanks also go to my friends, Mr. Chui Wai Yip, Miss Gladys Fan, Grace and Ling.

I would like to extend my appreciation and thanks for the financial support from the Australia Government for my research scholarship, OPRS. Thanks are also given to the friends who have made my life in Australia pleasant and unforgettable. Last, but by no means least, thanks to the Lord for all that He has done.

Seng Kah Phooi

Hobart, Monday 26th November, 2001.
School of Engineering, University of Tasmania,
TAS 7001, AUSTRALIA

List of Publications

1. S. K. Phooi, Man Zhihong, H.R. Wu, "Nonlinear adaptive RBF neural filter with Lyapunov adaptation algorithm and its application to nonlinear channel equalization", Proc. of the Fifth Int. Symposium on Signal Processing and its Applications, Brisbane, August, vol. 1 pp. 151-154, 1999.

2. Seng Kah Phooi, Zhihong Man, H.R. Wu, "Adaptive RBF filter with its application to nonlinear channel equalization", 2nd International Conference on Information, Communications and Signal Processing, ICICS'99, pp. paper no. 377, 1999.

3. Seng Kah Phooi, Zhihong Man, H.R. Wu, "Lyapunov stability-based RBF neural networks for nonlinear adaptive predictive coding", 2nd International Conference on Information, Communications and Signal Processing, ICICS'99, pp. paper no. 421, 1999.

4. S. K. Phooi, Man Zhihong, H.R. Wu, "A nonlinear Lyapunov RBF neural equalizer with its application to digital communication channel equalization", 4th IEEE Malaysia International Conference on Communication, MICC and ISCE'99, Melaka, September, vol. 1 pp. 235-240, 1999.

5. Man Zhihong, S. K. Phooi, H.R. Wu, "Lyapunov stability based adaptive backpropagation for discrete time system", Proc. of the Fifth Int. Symposium on Signal Processing and Its applications, Brisbane, August, vol. 1 pp. 661-664, 1999.

6. Man Zhihong, S. K. Phooi, H.R. Wu, "A new approach to chaotic based communication", 4th IEEE Malaysia International Conference on Communication, MICC and ISCE'99, Melaka, September, vol. 1 pp. 199-204, 1999.

7. Zhihong Man, Seng Kah Phooi, H.R. Wu, "Lyapunov stability-based adaptive backpropagation for discrete and continuous time systems", 2nd International Conference on Information, Communications and Signal Processing, ICICS'99, pp. paper no. 376, 1999.

8. Seng Kah Phooi, Zhihong Man, H.R. Wu, "Nonlinear active noise control using Lyapunov theory and RBF neural network", The 2000 IEEE Workshop on Neural Network for Signal Processing (NNSP'2000), vol 2, pp. 916-925, 2000.

9. Seng Kah Phooi, Zhihong Man, H.R. Wu, "A New Adaptive IIR Filter with Lyapunov Stability Theory" - The 7th IEEE Singapore International Conference on Communication Systems (ICCS), *CDROM.*

10. Seng Kah Phooi, Zhihong Man, W. Y. Chui, H.R. Wu "A New Computation Efficient Adaptive Algorithm for Parallel-Cascade Truncated Volterra System", The 4th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP2000).

11. Seng Kah Phooi, Zhihong Man, W. Y. Chui, H.R. Wu, "A New Concurrent Algorithm For Adaptive Filtering In Parallel Signal Processing ", The 4th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP2000).

12. Seng Kah Phooi, Zhihong Man, H.R. Wu, "Designing a fuzzy adaptive NLMS algorithm", International Conference on Artificial Intelligence in Science and Technology (AISAT'2000), pp. 39-43, 2000.

13. Seng Kah Phooi, Zhihong Man, H.R. Wu, "Complex RBF network with Lyapunov theory training algorithm", International Conference on Artificial Intelligence in Science and Technology (AISAT'2000), pp. 44-49, 2000.

14. Seng Kah Phooi, Zhihong Man, W. Y. Chui, H.R. Wu, "On-line Identification of nonlinear system using polynomial basis function neural network and Lyapunov theory", International Conference on Artificial Intelligence in Science and Technology (AISAT'2000), pp. 219-223, 2000.

15. Seng Kah Phooi, Zhihong Man, H.R. Wu, "A New Approach In Designing An Adaptive Lattice Predictor For Speech Signal In ADPCM" - accepted by the 8th Australian International Conference on Speech Science & Technology (SST-2000).

16. Seng Kah Phooi, Zhihong Man, H.R. Wu, "Nonlinear And Noisy Time Series Prediction Using A Hybrid Nonlinear Neural Predictor ", The Second International Conference On Intelligent Date Engineering And Automated Learning (IDEAL 2000)

17. Seng Kah Phooi, Zhihong Man, H.R. Wu, "Designing a fuzzy gain Lyapunov adaptive filter algorithm" - accepted and will be edited in the book in Springer-Verlag series "Studies in Fuzziness and Soft Computing".

18. Seng Kah Phooi, Zhihong Man, H.R. Wu, "A New Approach in Fuzzy Adaptive Filtering" - accepted and will be edited in the book in Springer-Verlag series "Studies in Fuzziness and Soft Computing".

19. Seng Kah Phooi, Zhihong Man, W. Y. Chui, H.R. Wu, "A New Computation Efficient Adaptive Algorithm for Parallel-Cascade Truncated Volterra System" - long paper published by World Scientific.

20. Seng Kah Phooi, Zhihong Man, W. Y. Chui, H.R. Wu, "A New Concurrent Algorithm For Adaptive Filtering In Parallel Signal Processing" - short paper published by World Scientific.

21. Seng Kah Phooi, Zhihong Man, H.R. Wu, "Nonlinear And Noisy Time Series Prediction Using A Hybrid Nonlinear Neural Predictor" - will be an edited in the book published by Springer-Verlag, Heidelberg and will be published in LNCS/LNAI series.

22. Seng Kah Phooi, Zhihong Man, H.R. Wu, "Designing a fuzzy gain Lyapunov adaptive filter algorithm", the Second International Discourse on Fuzzy Logic - With Fuzzy Logic In The New Millennium, pp. 47-49, 2000.

23. Seng Kah Phooi, Zhihong Man, H.R. Wu, "A New Approach in Fuzzy Adaptive Filtering", the Second International Discourse on Fuzzy Logic - With Fuzzy Logic In The New Millennium, pp. 50-53, 2000.

24. Zhihong Man, Seng Kah Phooi, H.R. Wu, "Further Investigation of Lyapunov Adaptive Filters", submitted to IEEE Transaction on Circuits & Systems, Part I. (REVISED)

25. Zhihong Man, Seng Kah Phooi, H.R. Wu, "Lyapunov Stability-Based Adaptive Backpropagation For Discrete-time Dynamical System ", submitted to IEEE Transactions of Neural Networks. (REVISED)

26. Seng Kah Phooi, Zhihong Man, W. Y. Chui, H.R. Wu, "Design of Adaptive Nonlinear Polynomial Filters Using Lyapunov Stability Theory ", submitted to Signal Processing, Elsevier Science.(REVISED)

27. Zhihong Man, Seng Kah Phooi, H.R. Wu, "Design of Adaptive Filter Using Lyapunov Stability Theory", submitted to IEEE Signal Processing Letters.

28. Seng Kah Phooi, Zhihong Man, H.R. Wu, "A New Adaptive IIR Filter In System Identification Using Lyapunov Stability Theory", submitted to Applied Signal Processing.

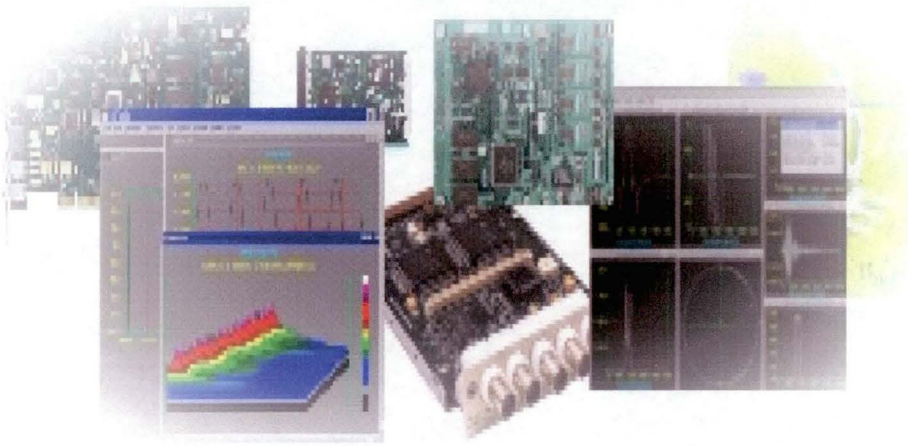
29. Seng Kah Phooi, Zhihong Man, H.R. Wu, "A Hybrid Nonlinear Neural Predictor And Its Application To Nonlinear And Noisy Time Series Prediction ", submitted to Neural Computing & Applications.

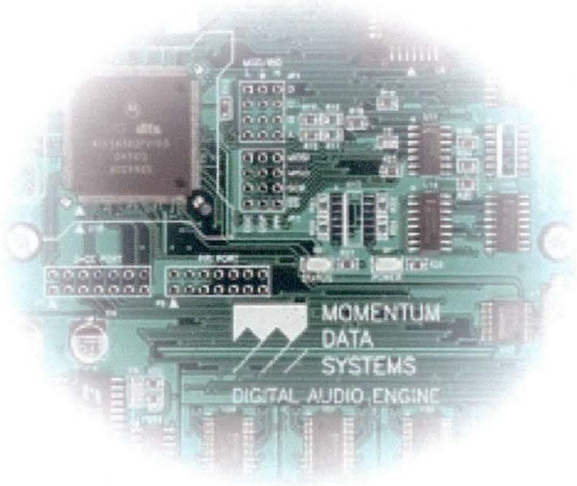
30. Seng Kah Phooi, Zhihong Man, H.R. Wu, "Lyapunov Stability-Based RBF Predictor For Adaptive Prediction of Nonlinear, Nonstationary Speech Signals In Adaptive Predictive Coding ", submitted to Neural Computing & Applications.

31. Seng Kah Phooi, Zhihong Man, H.R. Wu, "A New Approach In Feedforward Active Noise Control Using Lyapunov Stability Theory ", submitted to Adaptive Control and Signal Processing.

Contents

Chapter 1





Introduction

Chapter 1

Introduction

1.1 Motivation

The topic of adaptive filtering has matured to the point where it now constitutes an important part of signal processing. Whenever there is a requirement to process signals that result from operation in an environment of unknown statistics, the use of an adaptive filter offers an attractive solution to the problem as it usually provides a significant improvement in performance over the use of a fixed digital filter designed by conventional methods. In general, adaptive filtering can be defined as *performing some mapping from the input signal to the output signal with desired properties or adapting the filter response so as to make its output signal as close as possible in some sense to the desired response* [1]-[3]. It has been successfully applied in such diverse fields as communications, control, radar, sonar and seismology.

Adaptive filters generally consist of two distinct parts: a *filter*, whose structure is designed to perform a desired processing function, and an *adaptive algorithm* for adjusting the parameters of the filter. The many possible combinations of filter structures and the adaptive filtering algorithm governing them lead to a sometimes bewildering variety of adaptive filters. The adaptive filtering algorithm has a significant impact on the performance of the adaptive filtering. The performance of the adaptive algorithm is evaluated based on one or more of the following factors [3]: rate of convergence, stability, computational complexity, ability to track time varying characteristics, robustness to additive noise, numerical robustness and others. Ideally a low computational complexity and numerical robust adaptive filter with high rate

of convergence, high stability, fast tracking and robustness to additive noise properties is desired for many applications.

There are different approaches to the development of adaptive filter theory. The first approach is the approach based on *Wiener filter theory*. In this approach, a *tapped-delay line* or *transversal* filter as the structural basis for implementing the adaptive filter is used. Linear filters with a *finite impulse response* (FIR) are considered. Note that the filter output is a linear combination of a finite number of past inputs. The normal equation (i.e., the matrix equation defining the optimum Wiener solution) is modified by the use of the method of *steepest descent*, a well-known technique in optimization theory. The resulting algorithm is widely known as the *least mean square* (LMS) algorithm. In practice, the use of LMS is wide-spread due to its computational simplicity. Its major limitations are a relatively slow rate of convergence and its sensitivity to variations in the eigenvalue spread. Indeed, tracking will occur provided that the input data vary slowly compared to the learning rate of the LMS algorithm. The algorithm itself may be unstable if the learning rate of the LMS is not selected properly.

The second approach is based on *Kalman filter theory*. The Kalman filtering problem for a linear dynamic system is formulated in terms of two basic equations: the plant equation in terms of the state vector, and the measurement equation that describes measurement errors. Different recursive algorithms using the recursive solution to the Kalman filtering problem are derived. These algorithms can provide a faster rate of convergence than that attainable by the LMS algorithm and are insensitive to the eigenvalue-spread problem. Nevertheless, their limitation is high computational complexity.

The aforementioned two approaches are based on statistical concepts. The third approach is based on the classical method of *least squares*. It differs from these former two in that it is deterministic in its formulation. According to the method of least squares, the sum of weighted error squares is minimized. Different classes of adaptive filtering algorithms are derived. One of the classes is *recursive least-squares* (RLS) algorithm. This algorithm also assumes the use of a transversal filter as the structure basis of the adaptive filter. The derivation of the algorithm relies on

the matrix-inversion lemma. RLS algorithm has faster convergence rate and better tracking behaviour compared with LMS algorithm. However, it has suffered from the same limitation of high computational complexity) as the Kalman algorithm. To reduce the computational complexity, various modified RLS algorithms for adaptive filtering have been developed in [1],[7]-[16]. There are two families of modified RLS algorithms, corresponding to two possible filter structures: the *fast lattice* algorithms (FLA) [1],[12],[16] and the *fast transversal* filter (FTF) or *fast recursive least square* (FRLS) algorithms [1],[7],[8]. Although RLS algorithms exhibit fast convergence properties, they exhibit unstable performance [12]-[15]. Many modified schemes have been developed to improve the stability property [9],[12],[16],[17] but the stability problem of the adaptive filters has not been solved in the presence of noise or if there are some bounded input disturbances.

As aforementioned, the RLS and LMS algorithms are the popular algorithms for FIR filter. However, an *infinite impulse response* (IIR) filter can provide significantly better performance than a linear FIR filter having the same number of coefficients. This is a consequence of the output feedback that generates an infinite impulse response with only a finite number of parameters. A desired response can be approximated more effectively by the output of the IIR filter. Alternatively, to achieve a specified level of performance, an IIR filter generally requires considerably fewer coefficients. Fundamentally, there are two approaches to adaptive IIR filtering: *equation error* and *output error* methods. One major drawback of the *output error* IIR method is that unlike the adaptive FIR, the performance surface might not be hyper-paraboloid and thus has local minima to which the algorithm can converge [18]. Different adaptive algorithms [1]-[3],[18] based on the gradient search techniques in the adaptive IIR filter. However, after the cost function of the error is selected, the surface of the cost function in the parameter space is fixed. The search of the optimum parameters in the parameter space may stop at some local minimum because of the arbitrary initial condition of system states. Furthermore, the adaptive IIR filters have time varying poles and zeros, and the stability of the adaptive IIR filters using gradient search techniques may not be guaranteed.

Up to this point, the problems of the adaptive filtering algorithms in the adaptive filters have been noticed. These problems have motivated the research of new

adaptive algorithms that are less computational complexity, fast convergence, highly stability, fast tracking and robustness to additive noise. Besides adaptive filtering algorithms, the structure of the filter is another issue needed be considered. The common advantage of the adaptive FIR and IIR filters with linear model is their inherent simplicity but there are several circumstances that the performance of these linear model filter is unsatisfactory [74]. Therefore the development of nonlinear adaptive filtering would be desirable for many applications. One of the nonlinear model adaptive filters is the polynomial filter that includes *Volterra* and *Bilinear* filters. Besides polynomial filters, artificial intelligence techniques such as *neural networks* and *fuzzy logic* that have become recognized as powerful nonlinear approximation methods have also motivated the development of nonlinear adaptive filtering.

In summary, new techniques of adaptive filtering that can enrich the signal processing theory as well as have significant impact on effective and efficient filtering strategies have motivated the development of this thesis.

1.2 Scope

The aim of this thesis is to develop various new techniques of adaptive filtering using Lyapunov stability theory as first proposed in [19]. Reference [19] has only provided a basis foundation work on Lyapunov filtering. This thesis provides further work and analysis on the technique in [19]. Artificial intelligent techniques such as neural networks and fuzzy logic are also incorporated into the new filter designs. In line with this, a review of basis adaptive filtering techniques and adaptive algorithms is also provided in this thesis.

The essence of this thesis is the *Lyapunov stability-based adaptive filtering* (LAF) which introduces the *Lyapunov stability theory* [20] into the design of adaptive filters. This PHD research has established and consolidated the theoretical or mathematical foundation for Lyapunov stability-based adaptive filtering. The designed filter exhibits asymptotic convergence of the tracking error between the desired reference signal and the output of the adaptive filter. The filter also exhibits the stability and robustness with respect to the bounded disturbances such as additive

noises. This thesis has also provided a fundamental breakthrough in advancing the understanding of the stability, convergence and robustness of Lyapunov stability-based adaptive filtering algorithms. It has also pointed out that the Lyapunov stability-based adaptive filtering is an important subject in the area of signal processing. It has enriched the signal processing theory as well as has a significant impact on effective and efficient filtering strategies.

Adaptive filtering with linear models such as transversal FIR and IIR adaptive filters are first designed and analyzed. The resulted adaptive filtering algorithm is called *Lyapunov theory-based adaptive filtering* (LAF) algorithm. Nonlinear adaptive filtering techniques are also developed. These nonlinear methods include the design of *radial basis function* (RBF) neural network-based adaptive filtering with guaranteed Lyapunov stability. Fuzzy adaptive filters with Lyapunov sense fuzzy rules that allow linguistic information are also proposed. Another design of neural adaptive filters with Lyapunov sense backpropagation learning called *Lyapunov stability-based adaptive backpropagation* (LABP) is also developed. Both feedforward and recurrent neural networks are taken into account in these neural network designs. These nonlinear filtering schemes with neural networks and fuzzy logic are then followed by nonlinear polynomial filters such as *Volterra, Bilinear* and *parallel cascade Volterra* filters. Various adaptive filtering techniques based on the above methods are also developed. For example, a *concurrent Lyapunov theory-based adaptive filtering* (CLAF) algorithm is developed for the real-time implementation of large order filter when the computational time per iteration is critical. This is then followed by the *complex Lyapunov stability-based adaptive filtering* (Complex-LAF). A new *active noise control* (ANC) with a second path modeling scheme has been proposed. Two algorithms called *Filtered-X Lyapunov theory-based* algorithm (FXLYP), *Filtered-U Lyapunov theory-based* algorithm (FXLYP), and an overall on-line modeling techniques using the LAF are developed. A hybrid nonlinear filter that consists of nonlinear and linear sub-predictors are introduced. This predictor can be applied to several applications such as speech signal processing, financial time series etc. Difference schemes for system identification based on the aforementioned schemes are taken into consideration in this thesis. Other applications that are based on the work of this thesis can be found

in [51],[67],[111]-[116]. A review of the contents of the thesis is given in Section 1.3.

1.3 Thesis Outline

The organization of the thesis is as follows.

Chapter 1: This chapter introduces the major thrust of the thesis, giving the motivation, scope and thesis outline.

Chapter 2: This chapter constitutes mostly review material. It provides a background on existing techniques for adaptive filtering. An introduction of adaptive filter is presented. Various discussions such as the performance measures in adaptive filter and adaptive filtering system configurations are given subsequently. In the later part of the chapter, adaptive filter models such as *finite impulse response* (FIR) and *infinite impulse response* (IIR) are discussed. Adaptive algorithms for FIR and IIR filters are then reviewed.

Chapter 3: A new adaptive filtering technique called *Lyapunov Theory-based Adaptive Filtering* (LAF) is proposed in this chapter. A Lyapunov function of the error between the desired signal and the filter output is defined, the weights of the filter are then adaptively adjusted based on Lyapunov stability theory so that the error can asymptotically converge to zero. Unlike many adaptive filtering schemes using gradient search in the parameter space, the selected Lyapunov function for a Lyapunov filter has a unique global minimum in the state space. By properly choosing the parameter update law in Lyapunov sense, the output of the adaptive filter can asymptotically converge to the desired reference signal. Therefore, the local minima problem occurred in the gradient search-based adaptive filters is avoided. Although the input signal of the adaptive filter is disturbed by the bounded random noises, only the input and the output measurements are used for the design of the Lyapunov filters. Therefore, the design of Lyapunov adaptive filters is independent of the stochastic properties of the random input disturbances. It can be seen from the above discussion that Lyapunov stability theory provides an optimization method in the state space for the design of adaptive filters. In this chapter, we have also further

investigated the LAF filters by exploring the error convergence rate and the error convergence region in order to avoid the singularities.

Chapter 4: In this chapter, two realizations of the Lyapunov adaptive filters using *radial basis function* (RBF) neural networks are proposed. The FIR and IIR filters are configured as feedforward and recurrent RBF networks respectively. It is shown in Chapter 3 that a Lyapunov function of the error between the desired signal and the RBF neural network output is defined, the weights of the RBF neural filter are then adaptively adjusted based on the LAF in Chapter 3, so that the error can asymptotically converge to zero. Unlike many adaptive neural filtering schemes using gradient search in the parameter space, the selected Lyapunov function for the adaptive RBF filter has a unique global minimum in the state space. By properly choosing the weights update law in Lyapunov sense, the output of the adaptive RBF neural filter can asymptotically converge to the desired reference signal. Thus the local minima problem occurred in the gradient search-based adaptive filters is avoided,. Although the input signal of the RBF neural filter is disturbed by the bounded random noises, only the input and the output measurements are needed for the design of the RBF neural filters. Hence the proposed scheme is independent of the statistical properties of the input signals.

Chapter 5: The purpose of this chapter is to develop new kinds of nonlinear adaptive filters, which we refer to as *fuzzy adaptive filters.* First, a *fuzzy gain Lyapunov adaptive filter* for nonlinear adaptive filtering is proposed. This scheme is designed based on the LAF and fuzzy logic is introduced to the filter design. It incorporates fuzzy logic to the LAF by the use of a set of Lyapunov sense fuzzy if-then rules. Given the input signal and its squared norm, these rules are then used to determine the adaptive gain to update the filter parameters so that the error converges to zero asymptotically. The second fuzzy adaptive filter is named *LAF fuzzy adaptive filter.* This fuzzy adaptive filter is constructed from a set of changeable fuzzy IF-THEN rules. The LAF is used to update the parameter of the membership functions so that the dynamic error between the filter output and the desired response converges to zero asymptotically. Therefore, the most advantage of the fuzzy filter compared to the conventional filters is that linguistic information from human experts (in the form of fuzzy IF-THEN rules) can be incorporated into the filter. If no linguistic

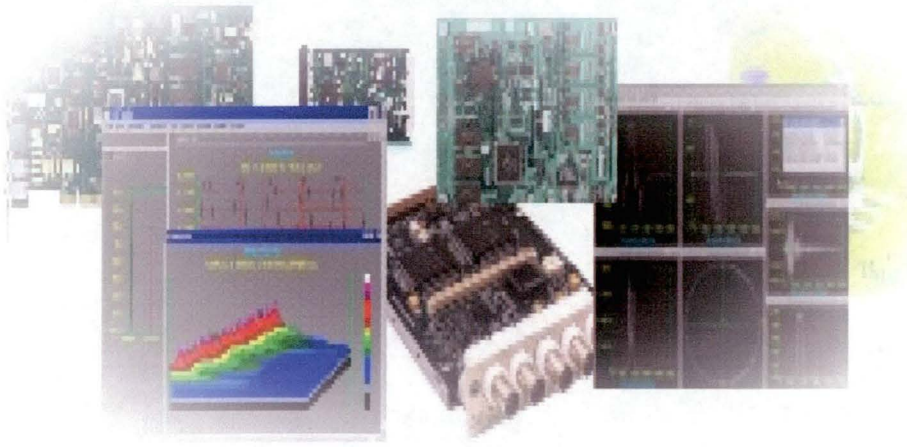
information is available, the fuzzy adaptive filters become well-defined nonlinear adaptive filters.

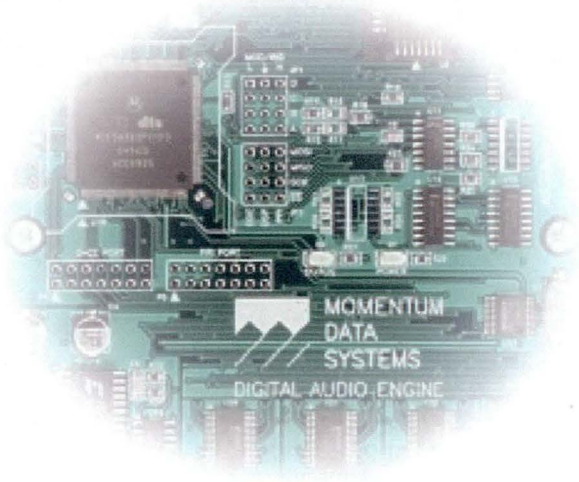
Chapter 6: A new approach of designing a BP algorithm using Lyapunov stability theory is proposed in this chapter. This chapter has also extended the ideals of *Lyapunov stability-based* algorithms in Chapter 3-5 to the design of the BP algorithm for *Time Delay Neural Network* (TDNN) with feedback in particular. We call this new algorithm as *Lyapunov Stability-based Adaptive Backpropagation* (LABP) algorithm. The designed LABP is a non-gradient based algorithm. In our new scheme, a Lyapunov function is defined for the error between the desired response and the neural network output. The defined criterion function is the Lyapunov function that has only unique minimum. The weights of neural network are then adaptively adjusted so that the error can converge to zero asymptotically. The network weights updated strategy is independent of signal statistical properties because only the desired response and input signal are required. The stability concern for the LABP algorithm is guaranteed by the Lyapunov Stability Theory.

Chapter 7: The objective of this chapter is to present one area of nonlinear signal processing known as polynomial signal processing using Lyapunov theory. The first part of this chapter presents a fast, low computation complexity and stable adaptive polynomial filters. We only focus on the following polynomial models: (1) *Volterra model* that the nonlinear system output signal can be related to the input signal through a truncated Volterra series expansion. (2) *Bilinear model* that involves and recursive nonlinear difference equation. The second part of the chapter considers another realization of nonlinear Volterra filter using *parallel-cascade* structure. Parallel-cascade realizations implement higher order Volterra systems as a parallel connection of multiplicative combinations of lower order truncated Volterra systems. All the proposed techniques in this chapter have excellent convergence and their stability are guaranteed by the Lyapunov stability theory. These schemes are independent of signals' stochastic properties. They have lower or comparable computational complexity compared to some conventional polynomial filters. Simulation examples have demonstrated the performance of these new designs.

Chapter 8: This chapter introduces other different techniques besides those proposed in Chapter 3-Chapter 6. These techniques include (1) A new concurrent algorithm for adaptive filtering called *concurrent Lyapunov theory-based adaptive filtering* (CLAF) in parallel signal processing. (2) Complex-valued Lyapunov theory-based adaptive filtering (Complex-LAF). (3) A new approach in feedforward active noise control using Lyapunov stability theory which consists two algorithms called *Filtered-X Lyapunov theory-based* algorithm (FXLYP), *Filtered-U Lyapunov theory-based* algorithm (FXLYP), and a overall on-line modeling techniques using the LAF. (4) A hybrid nonlinear neural predictor and its application to nonlinear and noisy time series prediction. Most of these methods are the modification of the scheme presented in Chapter 3-7 to suit particular applications.

Chapter 2





Adaptive Filtering

Chapter 2

Adaptive Filtering

2.1 Introduction

Filter is often used to describe a device that is applied to a set of noisy data in order to extract information about a prescribed quantity of interest. It can be used to perform three basic information-processing operations: *filtering, smoothing and prediction* [1]. It is linear if the filtered, smoothed or predicted quantity of interest at the output of the device is a linear function of the observations applied to the filter input. In the statistical approach to the solution of the linear filtering problems, we assume the availability of certain statistical parameters (i.e., mean and correlation functions) of the useful signal and unwanted additive noise and the requirement is to design a linear filter with the noisy data as input so as to minimize the effects of noise at the filter output according to some statistical criterion. A useful approach to this filter-optimization problem is to minimize the mean square value of the error signal that is defined as the difference between some desired response and the actual filter output. For stationary inputs, the resulting solution is commonly known as the *Wiener filter*, which is said to be optimum in the mean square sense. However, the Wiener filter is inadequate for dealing with nonstationary signal. Furthermore, the design of the Wiener filter requires a priori information about the statistics of the data. Therefore a more efficient method is to use an adaptive filter. The general structure of the adaptive filter is given in *Figure 2.1.*

An *adaptive filter* is a self-designing filter that relies for its operation on an adaptive algorithm, which makes it possible for the filter to perform satisfactorily in an environment where complete knowledge of the relevant signal characteristics is not

available. The algorithm starts from some predetermined set of initial conditions, representing complete ignorance about the environment. In a stationary environment, the algorithm converges to the Wiener solution in some statistical sense after successive iterations. In nonstationary environment, the algorithm can track time variations in the statistics of the input data, provided the variations are sufficiently slow. Therefore an adaptive filter is a *nonlinear device.* In another context, an adaptive filter is often refereed to as *linear* in the sense that the estimate of a quantity of interest is obtained adaptively (at the filter output) as a linear combination of the observations applied to filter input [1].

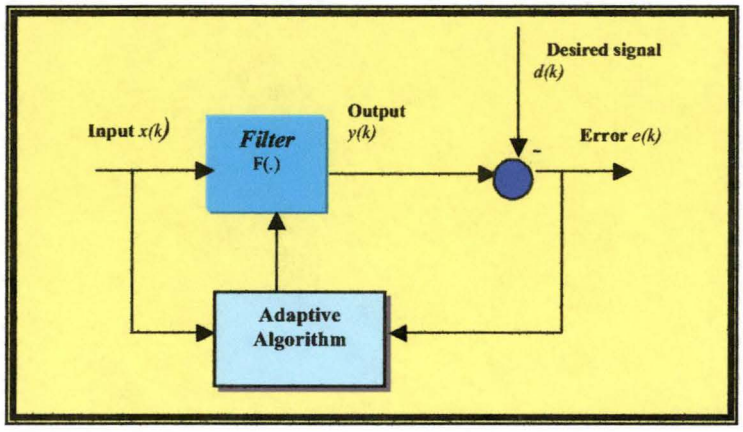


Figure 2.1: Block Diagram of the General Adaptive Filter

2.2 Performance Measures in Adaptive Filters

In designing an adaptive filter and its algorithm, there are a number of performance evaluations which are important. Thus six performance measures will be discussed in the following sections: convergence rate, minimum mean square error, computational complexity, stability, robustness, and filter length. [1]-[3]

Convergence Rate - The convergence rate determines the rate at which the filter converges to its resultant state. Usually a faster convergence rate is a desired characteristic of an adaptive system. Convergence rate is not, however, independent of all of the other performance characteristics. There will be a tradeoff, for example,

if the convergence rate is increased, the stability characteristics will decrease, making the system more likely to diverge instead of converge to the proper solution.

Minimum Mean Square Error - The minimum mean square error (MSE) is a metric indicating how well a system can adapt to a given solution. A small minimum MSE is an indication that the adaptive system has accurately modeled, predicted, adapted and/or converged to a solution for the system. [1]

Computational Complexity - Computational complexity is particularly important in real time adaptive filter applications. When a real time system is being implemented, there are hardware limitations that may affect the performance of the system. A highly complex algorithm will require much greater hardware as well as software resources than a simplistic algorithm. [1]

Stability - Stability is probably the most important performance measure for the adaptive system. By the nature of the adaptive system, there are very few completely asymptotically stable systems that can be realized. In most cases the systems that are implemented are marginally stable, with the stability determined by the initial conditions, transfer function of the system and the step size of the input. [1]

Robustness - The robustness of a system is directly related to the stability of a system. Robustness is a measure of how well the system can resist both input and quantization noise. [1]

Filter Length - The filter length of the adaptive system is inherently tied to many of the other performance measures. The length of the filter specifies how accurate a given system can be modeled by the adaptive filter. In addition, the filter length affects the convergence rate, by increasing or decreasing computation time, it can affect the stability of the system, at certain step sizes, and it affects the minimum MSE. [1]

In summary, ideally a low computational complexity, numerical robust adaptive filter with high rate of convergence, high stability, fast tracking and robustness to additive noise properties is desired for many applications. As in any engineering

problem, these desirable characteristics, in most cases, are incompatible with each other and some kind of trade-off is needed.

2.3 Adaptive Filtering System Configurations

There are four major types of adaptive filtering configurations: adaptive system identification, adaptive noise cancellation, adaptive linear prediction, and adaptive inverse system. All of the above systems are similar in the implementation of the algorithm, but different in system configuration. [3]

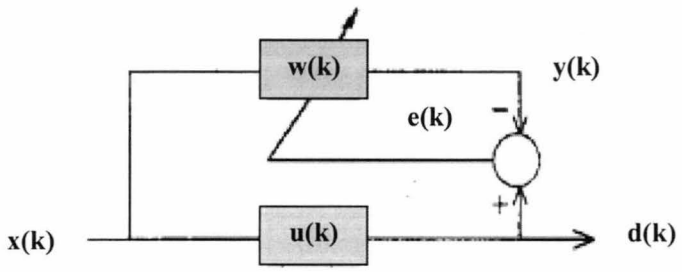


Figure 2.2

Adaptive System Identification Configuration [3] - The adaptive system identification is primarily responsible for determining a discrete estimation of the transfer function for an unknown digital or analog system. The same input *x(k)* is applied to both the adaptive filter and the unknown system from which the outputs are compared (*Figure 2.2*). The output of the adaptive filter *y(k)* is subtracted from the output of the unknown system resulting in a desired signal *d(k)*. The resulting difference is an error signal *e(k)* used to manipulate the filter coefficients of the adaptive system trending towards an error signal of zero. After a number of iterations of this process are performed, and if the system is designed correctly, the adaptive filter's transfer function will converge to, or near to, the unknown system's transfer function. For this configuration, the error signal does not have to go to zero, although convergence to zero is the ideal situation, to closely approximate the given system. Additionally the order of the adaptive system will affect the smallest error that the system can obtain.

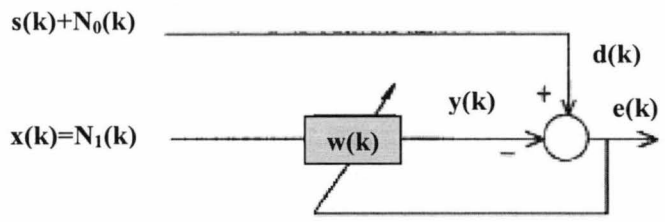


Figure 2.3

Adaptive Noise Cancellation Configuration [3] - In this configuration *(Figure 2.3),* the input *x(k)*, a noise source $N_1(k)$, is compared with a desired signal *d(k)*, which consists of a signal *s(k)* corrupted by another noise $N_0(k)$. The adaptive filter coefficients adapt to cause the error signal to be a noiseless version of the signal *s(k).* Both of the noise signals for this configuration need to be uncorrelated to the signal *s(k).* In addition, the noise sources must be correlated to each other in some way, preferably equal, to get the best results. Due to the nature of the error signal, the error signal will never become zero. The error signal should converge to the signal *s(k)*, but not converge to the exact signal.

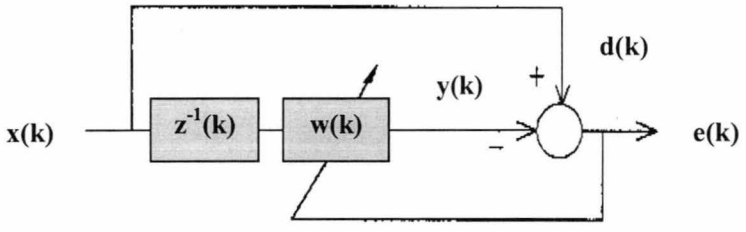


Figure 2.4

Adaptive Linear Prediction Configuration [3]- This configuration (*Figure 2.4*) essentially performs two operations. The first operation, if the output is taken from the error signal *e(k)*, is linear prediction. The adaptive filter coefficients are being trained to predict, from the statistics of the input signal *x(k)*, what the next input signal will be. As in the previous section, neither the linear prediction output nor the noise cancellation output will converge to an error of zero. This is true for the linear prediction output because if the error signal did converge to zero, this would mean that the input signal *x(k)* is entirely deterministic.

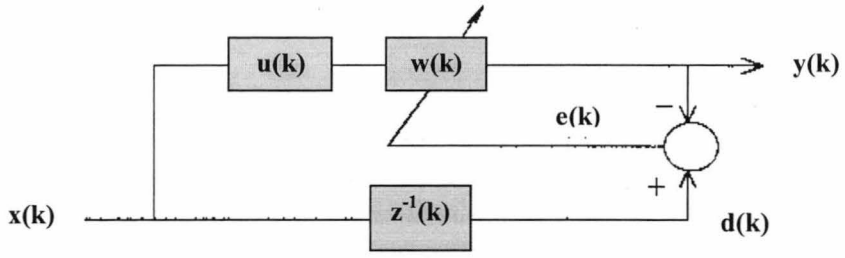


Figure 2.5

Adaptive Inverse System Configuration [3] - In this configuration (*Figure 2.5*). The goal of the adaptive filter here is to model the inverse of the unknown system *u(k)*. This is particularly useful in adaptive equalization where the goal of the filter is to eliminate any spectral changes that are caused by a prior system or transmission line. The way this filter works is as follows. The input *x(k)* is sent through the unknown filter *u(k)* and then through the adaptive filter resulting in an output *y(k)*. The input is also sent through a delay to attain *d(k)*. As the error signal is converging to zero, the adaptive filter coefficients *w(k)* are converging to the inverse of the unknown system *u(k)*.

2.4 Adaptive Filter Models

The FIR (*finite impulse response*) and IIR (*infinite impulse response*) are two popular linear models for adaptive filters. The FIR filter (also known as a tapped delay line or non-recursive filter), is the same as a *moving average* (MA) process. An infinite impulse response (IIR) filter has the same structure as an *autoregressive moving average* (ARMA) process.

A finite impulse response (FIR) filter is defined by [1]-[3]:

$$y(k) = \sum_{i=0}^{N-1} b_i(k)\,x(k-i) \tag{2.1}$$

An infinite impulse response (IIR) filter is defined by

$$y(k) = \sum_{i=0}^{N-1} b_i(k)x(k-i) + \sum_{i=1}^{N-1} a_i(k)y(k-i) \tag{2.2}$$

An IIR filter defined by (2.2) is referred to as having an *output error model* (OEM) where *y(k-i)* are past or feedback outputs. If the feedback *y(k-i)* is replaced by the desired response *d(k-i)*, the filter is said to have an equation error criterion, that is

$$y(k) = \sum_{i=0}^{N-1} b_i(k)x(k-i) + \sum_{i=1}^{N-1} a_i(k)d(k-i) \tag{2.3}$$

The equations (2.1) and (2.3) are linear in parameters and they have a quadratic performance surface with a unique minimum [18]. However, the equation error formulation may lead to biased parameters estimates [18]. On the other hand, the output error model is non-linear in the parameters and consequently the performance surface may be non-convex [18]. Thus, convergence to a local minimum is usually all that can be guaranteed using gradient search-based algorithms.

In the next section, we consider the adaptive algorithms in adaptive filters, focusing on *least mean square* (LMS) algorithm and *recursive least square* (RLS) algorithm specifically on adaptive FIR filters.

2.5 Adaptive Algorithms for Finite Impulse Response Filters

For FIR adaptive filters, the adaptive algorithms are linear-in-the-parameters, hence it is straightforward to find a convergent algorithm on the quadratic performance surface. Many algorithms exist, but we only consider the *least mean square* (LMS) and *recursive least square* (RLS) algorithms which are probably the most well known methods of adaptive algorithms.

2.5.1 Least Mean Square (LMS) Algorithm

Adaptive algorithms based on the gradient search or gradient descent adjust the filter parameters so as to move in the direction of the negative gradient of the performance criterion at the current point in parameter space. Gradient descent algorithms commonly use the mean square error performance surface (the LMS criterion) defined as

$$\varepsilon(k) = \frac{1}{2} E[e(k)^2] \tag{2.4}$$

The method of steepest descent follows the gradient towards the minimum, but at any instant it is not necessarily moving in the direction of the minimum, unless the gradient direction is aligned with one of the parameter axes [1]-[3]. One simple method of gradient descent is based on estimating the performance criterion $\varepsilon(k)$ at sample points in the weight space. The weights are then adjusted in the direction of the negative gradient until the performance criteria has reached the minimum. The general gradient descent methods is defined by

$$w(k+1) = w(k) - \mu \frac{\partial \varepsilon(k)}{\partial w} \tag{2.5}$$

where $w(k)$ is the filter parameter vector at time k, and μ is the *learning gain* or *step-size.*

The LMS algorithm can be defined as

$$w(k+1) = w(k) - \mu e(k)x(k) \tag{2.6}$$

where $e(k)$ is the error. [3]

The LMS algorithm is derived by using the instantaneous squared error as an estimate of the performance surface [1]-[3]. Since the *mean square output error* (MSOE) surface is quadratic, gradient descent algorithms will converge to a unique global minimum. It has been proven that the LMS algorithm will converge to the optimal Wiener solution, which is defined in discrete time as the solution w_0 which minimizes the mean square error criterion. The optimal Wiener solution is found in this case by solving the normal equation given by

$$\mathbf{R}w_0 = \mathbf{p} \tag{2.7}$$

where $\mathbf{R}$ is the correlation matrix of the input vector $X = [x(k), x(k-1), \ldots, x(k-n)]^T$, and $\mathbf{p}$ is the cross-correlation vector between X and the desired filter output $d(k)$ [1].

Convergence of the LMS Adaptive Filter

The convergence characteristics of the LMS adaptive filter is related to the autocorrelation of the input process as defined by

$$\mathbf{R}_x = E[\mathbf{x}(n)\mathbf{x}^T(n)] \tag{2.8}$$

There are two conditions that must be satisfied in order for the system to converge. These conditions include:

- The autocorrelation matrix, $\mathbf{R}_x$, must be positive definite.
- $0 < \mu < 1/\lambda_{max}$., where λ_{max} is the largest eigenvalue of $\mathbf{R}_x$.

In addition, the rate of convergence is related to the eigenvalue spread. This is defined using the condition number of $\mathbf{R}_x$, defined as $\kappa = \lambda_{max}/\lambda_{min}$, where λ_{min} is the minimum eigenvalue of $\mathbf{R}_x$. The fastest convergence of this system occurs when $\kappa = 1$, corresponding to white noise. This states that the fastest way to train the LMS adaptive system is to use white noise as the training input. As the noise becomes more and more coloured, the speed of the training will decrease.

Stability of the LMS Algorithm

It can be shown that starting with an arbitrary initial weight vector, the LMS algorithm will converge in the mean and will remain stable as long as the step size (2.6) is in the range

$$1 < \mu < \frac{2}{tap - input\ power} \tag{2.9}$$

[1], which is an easy boundary to calculate.

Within that margin, the larger μ is, the faster the convergence but the less the stability around the minimum value. On the other hand, the smaller μ is, the slower the convergence but will be more stable around the optimum value.

LMS Variants [1],[142]

There are many variants of the LMS algorithm, some of which are very useful and some are of little more than academic interest. A few of the more common ones are:

Block LMS [1],[142]- The weight vector of the FIR filter is held constant for a few iterations while an improved estimate of the performance surface gradient is obtained.

Variable Step Size [142] - The value of μ is chosen large at the beginning and then is progressively reduced to a smaller size to iterate closer to the optimum value.

Leaky LMS [1],[142] - This variation is addressed to systems with small wordlengths where round-off noise is fed back to adaptive weights and accumulates in time without bound leading to overflow. A small bias factor, b which is slightly less than one, is built in to bias each weight toward zero on each iteration counteracting the effect of noise build up:

$$w(k+1) = bw(k) - 2\mu e(k)x(k) \tag{2.10}$$

Sign Error LMS [142] - The computation needed by the adaptive algorithm can be reduced to zero multiplications and N additions using only the sign of the error signal (and making μ be a power of two):

$$w(k+1) = w(k) - \mu \text{ sign } [e(k)]x(k) \tag{2.11}$$

2.5.2 Recursive Least Square (RLS) Algorithm

The *quasi-Newton* adaptive algorithm uses second order statistics to improve the convergence rate of an adaptive filter, via the Gauss-Newton method. Probably the best known quasi-Newton algorithm is the *recursive least squares* (RLS) algorithm. It is important to note that even with the improvement in convergence rate, the RLS algorithm requires great amounts of processing power, which can make it difficult to implement on real-time systems.

There are a number of other quasi-Newton algorithms that have fast convergence rates, and that are also feasible alternatives for real time processing. See [1],[2] and [3] for more information.

That is, *w(k)* is filter coefficient vector and *u(k)* is input vector. Below are equations used for update of the filter coefficient vector.

$$g(k) = \frac{\lambda^{-1} P(k-1) X(k)}{1 + \lambda^{-1} X^T(k) P(k-1) X(k)} \tag{2.12}$$

$$\alpha(k) = d(k) - W^T(k) X(k) \tag{2.13}$$

$$W(k) = W(k-1) + g(k)\alpha(k) \tag{2.14}$$

$$P(k) = \lambda^{-1} P(k-1) - \lambda^{-1} g(k) X^T(k) P(k-1) \tag{2.15}$$

The RLS algorithm can be summarized by above 4 equations. Those equations are used to update the filter coefficients. *α(k)* is the priori estimation error. Equation (2.14) describes the adaptive operation of the algorithm, whereby the tap-weight vector is updated by incrementing its old value by an amount equal to the a priori estimation error *α(k)* times the time-varying gain vector *g(k)*. Equations (2.12) and (2.15) enable us to update the value of the gain vector itself. An important feature of the RLS algorithm described by these equations is that the inversion of the correlation matrix is replaced at each step by a simple scalar division. *P(k)* is the inverse of correlation matrix of input vector *X(k)*. Fast versions of the recursive algorithms for FIR filters have been proposed in numerous papers [1],[7],[8],[12],[16].

One problem that exists with RLS algorithms is that they exhibits unstable performance [12]-[15]. The exponential divergence of numerical errors was demonstrated in [12] for one particular signal. Furthermore, RLSs with forgetting also exhibit long time divergence under the impact of noise at the filter inputs [9]. Methods of avoiding instability have been proposed in [9],[12],[16],[17] but the stability problem of the adaptive filters have not been solved if there are some bounded input disturbances.

2.6 Adaptive Algorithms for Infinite Impulse Response Filters

The IIR filter is known to have better modelling properties than the FIR filter due to its rational (pole-zero) form. In comparison with the FIR filter, the IIR filter is nonlinear in the parameters, and so the parameters must be found by a recursive process. This has been the subject of much recent ongoing research [18],[117],[118]. An aspect of particular interest recently is the determination of conditions under which the LMS error criterion will have a unique global minimum [119]. In the following discussion, a review is given of the current state of the art in adaptive IIR algorithms and their convergence properties.

An off-line method of estimating the coefficient in an IIR filter was described by Steiglitz and McBride in 1965 [120]. This algorithm was analyzed by Stoica and Soderstrom, and an on-line version given [117].

Apparently the first *recursive prediction error method* (RPEM) for IIR adaptive filters was published in 1975 by White [125]. This method uses an mean square output error criterion and is based on finding the parameters which minimize the instantaneous squared error: the algorithm is described below. Consider first a direct form IIR filter described by

$$y(k) = \sum_{i=0}^{N-1} b_i(k)x(k-i) + \sum_{i=1}^{N-1} a_i(k)y(k-i) \tag{2.16}$$

The filter may be expressed as

$$y(k) = B^T(k)\tilde{X}(k) + A^T(k)\tilde{Y}(k-1) = \phi^T(k)\theta(k) \tag{2.17}$$

where $$A(k) = [a_1(k), a_2(k), \cdots, a_{N-1}(k)]^T \tag{2.18}$$

$$B(k) = [b_0(k), b_1(k), \cdots, b_{N-1}(k)]^T \tag{2.19}$$

$$\tilde{X}(k) = [x(k),\ x(k-1), \cdots,\ x(k-N+1)]^T \tag{2.20}$$

$$\tilde{Y}(k) = [y(k-1), y(k-2), \cdots, y(k-N+1)]^T \tag{2.21}$$

$$\theta(k) = [a_1(k), a_2(k), \cdots, a_{N-1}(k), b_0(k), b_1(k), \cdots, b_{N-1}(k)]^T \tag{2.22}$$

$$\phi(k) = [y(k-1), y(k-2), \cdots, y(k-N+1), x(k), x(k-1), \cdots, x(k-N+1)]^T \tag{2.23}$$

The algorithm developed by White uses the least mean-square error criterion

$$\varepsilon(k) = \frac{1}{2}E[e(k)]^2 = \frac{1}{2}E[(d(k) - y(k))]^2 \tag{2.24}$$

Since the true value $\varepsilon(k)$ is unknown, the coefficients are updated to minimize the instantaneous estimate of the expected error, $\varepsilon(k) = \frac{1}{2}e(k)^2$ [18]. Hence the parameters are updated according to

$$\Delta\theta = -\eta(k)\nabla_\theta \varepsilon(k) \tag{2.25}$$

where $$\nabla_\theta \varepsilon(k) = E[(d(k) - y(k)\left(-\frac{\partial y(k)}{\partial \theta}\right)] = -E[(d(k) - y(k)\frac{\partial y(k)}{\partial \theta})] \tag{2.26}$$

Hence

$$b_i(k+1) = b_i(k) + \eta e(k)\frac{\partial y(k)}{\partial b_i(k)}, \qquad i = 0, 1, \ldots, N\text{-}1 \tag{2.27}$$

$$a_i(k+1) = a_i(k) + \eta e(k)\frac{\partial y(k)}{\partial a_i(k)}, \qquad i = 1, \ldots, N\text{-}1 \tag{2.28}$$

where $$\frac{\partial y(k)}{\partial b_i(k)} = x(k-i) + \sum_{m=1}^{N-1} a_m(k)\frac{\partial y(k-m)}{\partial b_i(k)} \tag{2.29}$$

$$\frac{\partial y(k)}{\partial a_i(k)} = y(k-i) + \sum_{m=1}^{N-1} a_m(k)\frac{\partial y(k-m)}{\partial a_i(k)} \tag{2.30}$$

The simplifying assumption that $b_i(k-1) \approx b_i(k-2) \approx \ldots \approx b_i(k-m)$, and $a_i(k-1) \approx a_i(k-2) \approx \ldots \approx a_i(k-m)$, gives the following equation (recursive in the partial derivatives) [18],[118].

$$\begin{aligned} \frac{\partial y(k)}{\partial b_i(k)} &= x(k-i) + \sum_{m=1}^{N-1} a_m(k)\frac{\partial y(k-m)}{\partial b_i(k-m)} \\ &= \frac{1}{A(q^{-1})}x(k-i) \end{aligned} \tag{2.31}$$

$$\frac{\partial y(k)}{\partial a_i(k)} = y(k-i) + \sum_{m=1}^{N-1} a_m(k) \frac{\partial y(k-m)}{\partial a_i(k-m)}$$
$$= \frac{1}{A(q^{-1})} y(k-i) \tag{2.32}$$

The filtering imposed by the $\frac{1}{A(q^{-1})}$ term on the data is characteristic of the IIR filter algorithm. Since this filtering operation takes place for each element of the data vector, the amount of storage required is $O(2N^2-4N+2)$. this algorithm is also known as *Stearns algorithm* [117],[121].

White's algorithm requires a significant amount of storage, and so a simplified gradient algorithm requiring only one regressor filter was proposed by Hsia [137]. In this case, the gradient terms (regressors in the update equation), are filtered first (via the $\frac{1}{A(q^{-1})}$) autoregressive filter) before being passed through the delays. Thus, the filter has $\frac{x(k)}{A(q^{-1})}$ and $\frac{y(k)}{A(q^{-1})}$ terms being delayed. This differs from White's algorithm where a separate filter is employed after each delay in the input and output of the model. Thus (2.31) and (2.32) are replaced by

$$\frac{\partial y(k)}{\partial b_i(k)} = \frac{x(k)q^{-i}}{A(q^{-1})|_{k-i}} = x(k-i) + \sum_{m=1}^{N-1} a_m(k-i) \frac{\partial y(k-m)}{\partial b_i(k-m)} \tag{2.33}$$

$$\frac{\partial y(k)}{\partial a_i(k)} = \frac{y(k)q^{-i}}{A(q^{-1})|_{k-i}} = y(k-i) + \sum_{m=1}^{N-1} a_m(k-i) \frac{\partial y(k-m)}{\partial a_i(k-m)}$$
$$= \frac{1}{A(q^{-1})} y(k-i) \tag{2.34}$$

where $A(q^{-1})|_{k-p}$ is defined as

$$A(q^{-1})|_{k-p} \equiv q^{-p} - \sum_{i=1}^{N-1} a_i (q^{-i-p}) \tag{2.35}$$

Hsia notes that the transient performance of the algorithm may be different to White's, due to the differences between $\theta(k)$ and $\theta(k-i)$,and this will be more pronounced for large i. The modified algorithm introduces by Hsia gives very similar result to White's algorithm in steady state, (but may deviate during the transient

phase of weight adaptation), with a significant reduction in memory requirements, and computational complexity. The storage for Hsia's algorithm is *O(3N-3)*, while the computational complexity is reduced from $O(2N^2-4N+2)$ *to* $O(N-1)$.

Feintuch [122] presented a simplified version of the White's algorithm by approximating (2.31) and (2.32) as

$$\frac{\partial y(k)}{\partial b_i(k)} \approx x(k-i) \tag{2.36}$$

$$\frac{\partial y(k)}{\partial a_i(k)} \approx y(k-i) \tag{2.37}$$

That is, ignores dependence on past data. It was shown that Feintuch's algorithm may not converge to any minimum on the *mean square output error* (MSOE) surface unless a *strictly positive real* (SPR) condition is satisfied [18],[123]. This algorithm is also known as a *pseudo-linear regression* (PLR) due to the fact that the nonlinear regression ignores the dependence on past parameters [18]. Another example of PLS is the extended least squares (ELS) algorithm discussed in [124].

Instrumental variable methods are not influenced by a multimodel error surface because the algorithm is not minimizing the instantaneous error. In a *prediction error model* (PEM) the noise is modelled, as opposed to an OEM model [126]. IVM and PEM tend to result in more accurate models than the OEM. The OEM can be regarded as a deterministic model in the sense that noise is not modelled. Friedlander presented a *recursive maximum likelihood* (RML) algorithm for IIR filter in [129]. The algorithm is not based on gradient descent, but rather depends on a least-squares solution. A signal model is used indirectly to estimate the filter coefficients.

Fan and Jenkins introduced a set of algorithms for the IIR filter which are based on the *Steglitiz-McBride* method [117]. An important point to note with these algorithms, is that they do not minimize the MSOE, and consequently are not affected by local minima in the surface [127]. A convergence proof for a version of he algorithm using stochastic methods, and relating it to an associated *ordinary differential equation* (ODE) [124], was presented in [128].

Johnson developed IIR adaptive algorithm by considering the coefficient update procedure as a linear system with time-varying nonlinear feedback [130],[131]. This algorithm is known as the *hyperstable adaptive recursive filter* (HARF) since hyperstability theory was applied to derive the weight update equations. Previously this had been used on time-varying nonlinear control systems [132]. A linear time-invariant system $G(q) = \frac{D(q)}{C(q)}$ is hyperstable, if for any input *x(k)*, and output *y(k)*

$$\sum_{k=0}^{\tau} y(k)x(k) < K^2 \qquad \forall\ K\ (k = 0) \tag{2.38}$$

This implies that the system is bounded for any input *x(k)*.

The HARF algorithm is dependent on an auxiliary model of the IIR filter given by

$$f(k) = \sum_{i=1}^{N-1} a_i(k+1)f(k-i) + \sum_{i=0}^{N-1} b_i(k+1)x(k-i) \tag{2.39}$$

The actual filter output is given by

$$y(k) = \sum_{i=1}^{N-1} a_i(k)f(k-i) + \sum_{i=0}^{N-1} b_i(k)x(k-i) \tag{2.40}$$

The update equations are updated according to

$$a_i(k+1) = a_i(k) + \frac{\mu_i(k)}{\rho(k+1)} f(k-i)v(k) \qquad 1 \geq i \geq N-1 \tag{2.41}$$

$$b_i(k+1) = b_i(k) + \frac{\nu_i(k)}{\rho(k+1)} f(k-i)v(k) \qquad 0 \geq i \geq N-1 \tag{2.42}$$

where μ_i and ν_i are positive constants, $\rho(k)$ is a normalization factor given by

$$\rho(k) = 1 + \sum_{i=1}^{N-1} \mu_h f^2(k-h) + \sum_{i=0}^{N-1} \nu_h x^2(k-h) \tag{2.43}$$

and the regression signal *v(t)* is given by

$$v(k) = (d(k) - y(k)) + \sum_{l=1}^{M} s_l(d(k-l) - f(k-l)) \tag{2.44}$$

The constants *M*, and s_l are chosen so that *G(q)* is strictly positive real (SPR), where

$$G(q) = \frac{1 + \sum_{l=1}^{M} s_l(q^{-l})}{1 - \sum_{l=1}^{N-1} a_l(q^{-l})} \tag{2.45}$$

The SPR condition must hold for a system to be hyperstable, which implies [131]:

$$\mathrm{Re}[H(q)] > 0 \qquad q=e^{j\theta} \tag{2.46}$$

Larimore et al developed a simplified HARF (SHARF) algorithm which is convergent for slow rates of adaptation [131]. The SHARF algorithm is equivalent to a filtered error algorithm [18]., and is derived by noting that when the learning rates μ_i and ν_i are small, $a(k+1) \approx a(k)$, $b(k+1) \approx b(k)$, and $f(k) \approx d(k)$. Thus, the auxiliary process in (2.39) is no longer required. The regressor $v(k)$ can be simplified to

$$\begin{aligned} v(k) &= (d(k) - y(k)) + \sum_{l=1}^{M} s_l(d(k-l) - y(k-l)) \\ &= e(k) + \sum_{l=1}^{M} d_l e(k-1) \end{aligned} \tag{2.47}$$

The update equations for the weights are

$$a_i(k+1) = a_i(k) + \mu_i v(k) d(k-i) \tag{2.48}$$

$$b_i(k+1) = b_i(k) + \nu_i v(k) x(k-i) \tag{2.49}$$

It can be seen that these update equations are the same as Feintuch's algorithm when $M=0$.

Convergence for the SHARF algorithm will occur provided learning rates are small, and the SPR condition holds for ([131]):

$$G(q) = \frac{1}{1 - \sum_{l=1}^{N-1} a_l(q^{-l})} \tag{2.50}$$

White's algorithm is classified as a recursive predicted method (RPEM) which has a filtered regression vector, while HARF, SHARF, and Feintuch's algorithm are classed as *pseudolinear regression* (PLR) algorithm since the filtered regression vector is simplified to be the data vector [124]

The recursive maximum likelihood (RML) and other algorithms are presented in the unified framework by Ljung in [124]. Friedlander observed poor learning

performance of the recursive maximum likelihood estimator when modeling ARMAX systems given by

$$y(k) = \frac{B(q^{-1})}{A(q^{-1})}u(k) + \frac{C(q^{-1})}{A(q^{-1})}v(k) \qquad (2.51)$$

where the roots of the $C(q^{-1})$ polynomial are near the unit circle, where $u(k)$ is the input, and $v(k)$ is a sequence of independent random values.

Friedlander proposed an improvement to RML methods in [134], where a prefilter is used, which pulls the roots further into the unit circle. Friedlander conjectured that the use of a prefilter which pulled the roots closer to the origin would reduce the response time of the parameter changes to prediction errors, hence resulting in faster convergence times. Simulation results confirmed this.

Murali and Rao, applied this prefiltering scheme to Hsia's modified gradient algorithms [122],[135]. Similar improvements in performance were observed in this case also. The algorithm is modified by replacing $A(q^{-1})$ with $D(q^{-1})$ where

$$D(q^{-1}) = A(cq^{-1}) \qquad (2.52)$$

$$= 1 - \sum_{i}^{N} a_i(k)(cq^{-i}) \qquad 0 \le c \le 1 \qquad (2.53)$$

where c is a pulling factor. As c decreases, the roots of $A(q^{-1})$ move radially inwards towards the origin [134].

Ljung gave results for the convergence of algorithms of this type [136] which depends on the SPR condition of the transfer function

$$G(q^{-1}) = \frac{D(q^{-1})}{\underline{A}(q^{-1})} - \frac{1}{2} \qquad (2.54)$$

where $\underline{A}(q^{-1})$ is the polynomial of true parameters and determines the convergence of the algorithm.

2.6.1 Convergence and Error Surface of Adaptive IIR filters

Analysis of these algorithms in terms of convergence and transient performance is important in order to obtain an understanding of its behaviour. It is normally desired to do this theoretically, although simulations are also useful to obtain an indication of the practical performance. This is particularly true for algorithms with complex behaviour.

Ljung has given a convergence analysis of a general recursive algorithm using the ODE method [124]. An important assumption that is used in his work that does not apply in the above algorithms is the assumption of decreasing gain. This implies that the convergence analyses performed by Ljung are not directly applicable here, unless the gains are made to approach zero. This is not desired from a real-time system identification point of view, because the unknown system may be time-varying, and therefore the algorithm should be able to track these parameter variations.

Other convergence methods have been proposed, including using Lyapunov functions [124], local linearization [2].

A number of researchers considered the *mean square output error* (MSOE) surface of the RPEM model, that is

$$y(k) = \frac{B(q^{-1})}{A(q^{-1})} x(k) \tag{2.56}$$

is the model, where no noise is considered, as in the *prediction error model* (PEM) [133].

Stearns was responsible for initially conjecturing that the error surface of an IIR filter would be unimodal, if the model order was sufficient, and it was fed with a white noise input.

Parikth and Ahmed studied the convergence properties of Stearn's algorithm. They showed that it converged to a local or global minimum mean-square-error value, depending on the initial weight values [121]. A sequential regression algorithm

which uses correlation information was introduced by Parikh and Ahmed and was observed to converge to a global minimum whereas Stearn's algorithm did not [139].

Soderstrom and Stoica in [126] proved this for the condition that $\hat{n_b} \geq n_a - 1$ and the input is white. This was showed to be false for filters of order higher than two, ie. $n_a > 2$, by Fan and Nayeri [119], while for orders less than or equal to two, the conjecture holds regardless of whether the condition imposed by Soderstrom and Stoica is met. For filters of higher order, Fan and Nayeri showed that Stearns conjecture does not hold (that is, the surface may not be unimodal), if the conditions of Soderstrom and Stoica are not met. Note that the condition is sufficient condition for uniqueness of the MSOE estimate, and hence the unimodality of the MSOE surface for an exact order model.

In the case where the MSOE is multimodal, convergence to a global minimum using gradient descent methods is not guaranteed. In this case, the starting point for the weights determines the convergence point of the algorithm. Other algorithms of reaching the global minimum have been devised which do not depend on the MSOE surface, and therefore it makes no difference whether the surface is multimoadel or not. Soderstrom and Stoica in [127] noted that because the Steiglitz-McBride Method (SMM) and *Instrumental Variable Method* (IVM) are not minimizing the MSOE, multimodality of the MSOE surface should not cause difficulties with either of these methods. Interesting results however, were shown by Fan and Nayeri on the convergence of the SMM, which may converge along the path of global minimum, but may abruptly move to converge to a different minimum point.

An IIR algorithm which uses prefiltering to achieve convergence to the global minimum was developed by Fan and Jenkins [138]. This algorithm is based on the SMM, and a convergence proof for the algorithm was given in [117].

Ljung proved the convergence of a general model encompassing the model used by White [141] using the ODE method [124]. This approach is based on formulating a differential equation for recursive update equation and examining the solution trajectories. The difference between Ljung's algorithm and that proposed by White, is that Ljung used a decreasing gain, whereas White used a fixed gain.

2.6.2 Stability of IIR Filter

Stability is a problem for adaptive IIR systems, since, during adaptation, if one of the poles of the characteristic equation $A(q^{-1}) = 0$ moves outside the unit circle, the filter becomes unstable. A number of methods exist to overcome these stability problems in IIR filters. The following stability tests are outlined in [18], and are described below.

For low order systems, the test $\sum_i |a_i| < 1$ will indicate whether a filter is stable. The problem with this method is that it is too restrictive, since clearly it will give false indications of instability. The modified Schur-Cohn test [140], indicates the presence of unstable poles without the restrictions above. It does not show which coefficients are responsible for the unstable pole, the polynomial needs to be factored to discover this. These are several methods of removing unstable poles. The first is to ignore the previous update step, and carry on. This may work, but there is the problem that if may become stuck in some way [118].

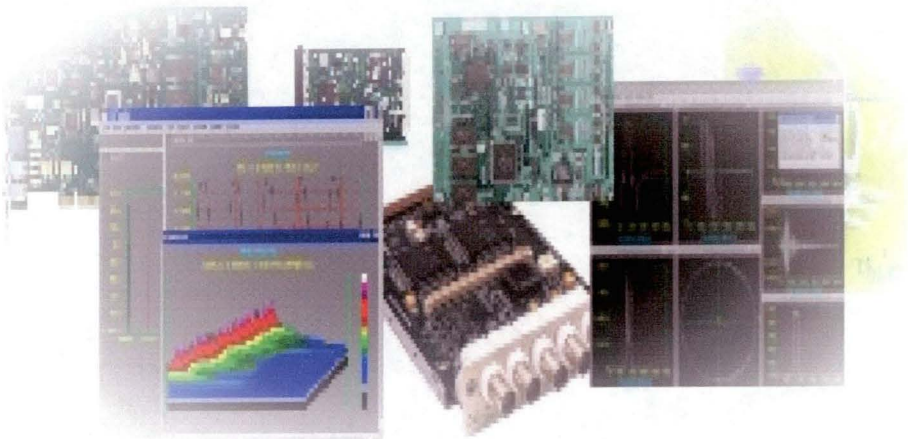
Another method is to factorize the $A(q^{-1})$ polynomial and use a projection method, the projection method involves computing the pole of the characteristic equation at each update, and if they fall outside the unit circle, to project them inside the unit circle. Performing this operation does not guarantee convergence to the desired parameter value, neither is there a proof that the projection will only be needed finitely many times [118]. A modification of this technique is to reduce the update step size until new parameter estimates do not become unstable [118].

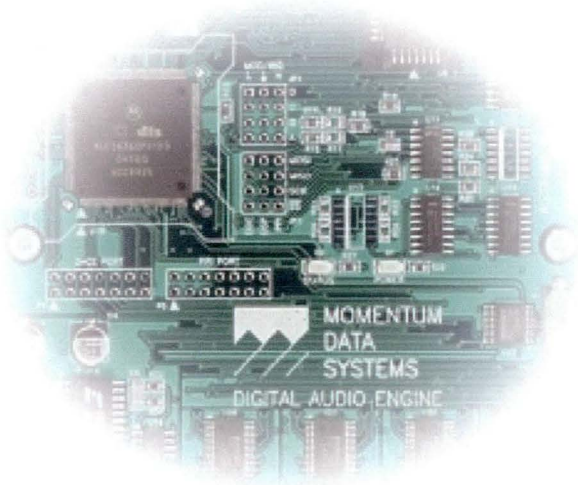
2.7 Conclusion

It is clear that there is a significant number of adaptive algorithms for FIR and IIR adaptive filtering. This chapter has considered the linear models and their associated algorithms. We have discussed several adaptive algorithms for FIR filter and their problems. At the same time, we have also explored there are more problems for IIR filter including the convergence to local minima and the filter stability. As shown, analysis of adaptive IIR algorithms is the subject of continuing research, aimed at establishing results for conditions of convergence and stability. In summary, a new adaptive algorithm that is less computational expensive, and numerically robust with high rate of convergence, high stability, fast tracking and robustness to additive noise properties is desired for many applications.

Although linear models perform well in adaptive filtering including system identification provided the basic assumptions about the system or signal are met. If a system or signal has nonlinear characteristics, then poor performance may be expected. Due to this reason, several new schemes which are based on nonlinear models will be presented in the later chapters of this thesis.

Chapter 3





Lyapunov Theory-based Adaptive Filtering

Chapter 3

Lyapunov Theory-based Adaptive Filtering

3.1 Introduction

Adaptive filtering has a wide variety of applications in adaptive control, high-resolution spectrum estimation, echo cancellation and channel equalization, system identification in general, adaptive differential encoding, interference suppression, adaptive deconvolution, biomedical signal processing, automatic process fault diagnosis, and many other fields [1]-[6]. The adaptive filtering problem has been described in Chapter 2. One of the simplest class of filter structure is linear filters with a *finite impulse response* (FIR). A typical FIR filter implemented in transversal structure is depicted in the Chapter 2.

There are two widely employed adaptive algorithms for FIR filter: *Least mean square* (LMS) [1]-[3] and *Recursive least squares* (RLS) algorithms [1]-[3]. The LMS [1]-[3] algorithms attempt to minimize a quadratic performance function by employing a stochastic gradient technique. This technique involves an instantaneous estimate of the gradient. The convergence of LMS algorithm is strongly dependent on the spectral characteristic of the input signal. The requirement of convergence imposes a bound on the gain of the LMS and this bound depends on the eigen-value spread of the autocorrelation matrix of the input signal. In practice, the use of LMS is wide-spread due to its computational simplicity.

Recursive least squares (RLS) algorithm [1]-[3] is another much-studied algorithm for FIR filters. This algorithm also assumes the use of a transversal filter as the

structure basis of the adaptive filter. The advantages of RLS over LMS in aspects such as tracking behavior and fast convergence are well known. It can be shown [1] that the convergence behavior of the RLS is independent of the spectral characteristics of the input signal. On of the drawback of RLS algorithm is its high computational complexity. For computational simplicity, various famous modification of RLS algorithms for adaptive filtering have been developed in [1],[7]-[16]. Some fast RLSs have been introduced that circumvent the computational burden of the Riccati equation in the conventional RLS. There are two families of such fast algorithms, corresponding to two possible filter structures: the *fast lattice* algorithms (FLA) [1],[12],[16] and the *fast transversal* filter (FTF) or *fast recursive least square* (FRLS) algorithms [1],[7],[8]. Note that different RLS algorithms (in transversal filter form) only differ in the way they compute a certain quantity often called the adaptation gain in general.

Although RLS algorithms exhibit fast convergence properties, they exhibit unstable performance [12]-[15]. The exponential divergence of numerical errors was demonstrated in [12] for one particular signal. Authors in [12] also reported the weighting parameter sensitive divergence on algorithm's long term. By the over-weighting of recent data at filter input, this parameter introduces algorithm's 'forgetting' that, when once set and kept as a constant during algorithm's run, can be regarding as a 'blind forgetting' case. Furthermore, RLSs with forgetting also exhibit long time divergence under the impact of noise at the filter inputs [9]. Methods of avoiding instability have been proposed in [9],[12],[16],[17] but the stability problem of the adaptive filters have not been solved if there are some bounded input disturbances.

Another realization of adaptive linear filter is *infinite impulse response* (IIR) filter. The IIR filter can provide significantly better performance than FIR filter having the same number of coefficients. This is a consequence of the output feedback that generates an infinite impulse response with only a finite number of parameters. Fundamentally, there have been two approaches to adaptive IIR filtering: *equation error* and *output error* methods. One major drawback of the *output error* method is that the performance surface might not be hyper-paraboloid and thus has local minima to which the algorithm can converge. Different adaptive algorithms [1]-

[3],[18] based on the gradient search techniques in the adaptive IIR filter. However, after the cost function of the error is selected, the surface of the cost function in the parameter space is fixed. The search of the optimum parameters in the parameter space may stop at some local minimum because of the arbitrary initial condition of system states. Another disadvantage of IIR system is that the adaptive IIR filters have time varying poles and zeros, and the stability of the adaptive IIR filters using gradient search techniques may not be guaranteed. Furthermore, if the disturbances are random signals, the mathematics of stochastic processes must be used for the optimization and parameter design.

While the adaptive filters are widely used for signal processing, the aforementioned convergence, stability, complexity and local minimum problems have been observed. To overcome these problems, a new adaptive filtering technique called *Lyapunov Theory-based Adaptive Filtering* (LAF) [19] is presented in this chapter. It is shown in [19] that a Lyapunov function of the error between the desired signal and the filter output is defined, the weights of the filter are then adaptively adjusted based on Lyapunov stability theory so that the error can asymptotically converge to zero. Unlike many adaptive filtering schemes using gradient search in the parameter space, the selected Lyapunov function for a Lyapunov filter has a unique global minimum in the state space. By properly choosing the parameter update law in Lyapunov sense, the output of the adaptive filter can asymptotically converge to the desired reference signal. Therefore, the local minima problem occurred in the gradient search-based adaptive filters can be avoided. Although the input signal of the adaptive filter is disturbed by the bounded random noises, only the input and the output measurements are used for the design of the Lyapunov filters. Therefore, the design of Lyapunov adaptive filters is independent of the stochastic properties of the random input disturbances. In addition, because the error dynamics of Lyapunov filters asymptotically converges to zero, the stability of both Lyapunov adaptive FIR and IIR filters is guaranteed. It can be seen from the above discussion that Lyapunov stability theory provides an optimization method in the state space for the design of adaptive filters.

In this chapter, we have further investigated the LAF filters by exploring the convergence rate of the error between the desired reference signal and the output of

the Lyapunov filter. We have discussed the convergence region of the error for the modified Lyapunov filter in order to avoid the singularities. These convergence properties are very useful to evaluate the performance of Lyapunov adaptive filters and to design the adaptive laws for practical application. A few simulation examples are performed to demonstrate the robustness and effectiveness of the Lyapunov FIR and IIR filters compared with a few gradient search-based adaptive filters.

The chapter is organized as follows. In section 3.2, the adaptive filtering strategy using Lyapunov stability theory [19] is presented. In section 3.3, the design of the Lyapunov theory-based adaptive filtering (LAF) is presented. The convergence rate of the Lyapunov filters is analyzed, and the convergence region of the modified Lyapunov adaptive filter to avoid the singularities is obtained. These analyses are presented in section 3.4. Section 3.5 extends the idea of LAF to IIR filters In section 3.6, simulation results are presented to show the good performance of the Lyapunov adaptive FIR and IIR filters.

3.2 Lyapunov Theory-based Adaptive Filtering (LAF) for FIR Filter

Unlike many adaptive filtering schemes using gradient search in the parameter space, the presented LAF algorithm uses a Lyapunov function $V(k)$, which is positive definite, with a unique global minimum in the state space [19]. By properly choosing the parameter update law in the sense that $\Delta V(k) = V(k) - V(k-1)$ is negative, the output of the adaptive filter can asymptotically converge to the desired reference signal according to Lyapunov stability theory [20]. Therefore, the local minima problem that occurs in the gradient search-based adaptive filters is avoided and the stability of the error dynamics are guaranteed at the same time.

The basic principle of the LAF filtering can be briefly introduced as follows. If the adaptive filter is implemented using FIR structure, it can be considered as moving average or MA model, in which the filter has only zeros, characterized by the difference equation

$$y(k) = \sum_{i=0}^{N-1} h_i(k)x(k-i) \tag{3.1}$$

The difference equation in (3.1) can be rewritten in vector form as

$$y(k) = H^T(k)X(k) \tag{3.2}$$

where $H(k) = [h_0(k), \ h_1(k), \ ..., \ h_{N-1}(k)]^T$

$X(k) = [x(k), x(k - 1), ..., x(k-N+1)]^T$

$X(k)$ is the input signal vector of the filter, which has been disturbed by the nonlinearity of the communication channel and noises. The $y(k)$ is the output of the filter, and $d(k)$, the *desired response*, is provided for the output of the filter to follow. The $e(k)$ is the error between the desired reference signal $d(k)$ and the output of the filter $y(k)$.

$$e(k) = d(k) - y(k) \tag{3.3}$$

The filter coefficient vector update equation is similar to RLS algorithm

$$H(k) = H(k - 1) + g(k)\alpha(k) \tag{3.4}$$

where $g(k)$ is the adaptation gain and $\alpha(k)$ is the a priori estimation error defined as

$$\alpha(k) = d(k) - H^T(k-1)X(k) \tag{3.5}$$

The adaptation gain $g(k)$ in (3.4) is adaptively adjusted using Lyapunov stability theory as (3.6) so that the error in (3.3) asymptotically converges to zero.

$$g(k) = \frac{X(k)}{\| X(k) \|^2}\left(1 - \kappa\frac{|e(k-1)|}{|\alpha(k)|}\right) \tag{3.6}$$

where $0 \leq \kappa < 1$. The circumstantial derivation and design of the LAF algorithm will be presented in next section

3.3 Design Of Lyapunov Theory-based Adaptive Filtering Algorithm Using Lyapunov Stability Theory

The design of the Lyapunov FIR filter is described in the following theorem [19]:

Theorem 3.1: For the given desired response *d(k)*, if the weight vector *H(k)* of the filter $y(k) = H^T(k)X(k)$ is updated as follows

$$H(k) = H(k-1) + g(k)\alpha(k) \tag{3.7}$$

and $$g(k) = \frac{X(k)}{\| X(k) \|^2}\left(1 - \kappa\frac{|e(k-1)|}{|\alpha(k)|}\right) \tag{3.8}$$

where *g(k)* is the adaptation gain, $\alpha(k)$ is the a priori estimation error defined as

$$\alpha(k) = d(k) - H^T(k-1)X(k) \tag{3.9}$$

and $$0 \le \kappa < 1, \tag{3.10}$$

then the error $e(k) = d(k) - y(k)$ asymptotically converges to zero.

Proof: Define a Lyapunov function

$$V(k) = e^2(k) \tag{3.11}$$

Then, we have $\Delta V(k) = V(k) - V(k-1)$

$$= e^2(k) - e^2(k-1)$$

$$= (d(k) - H^T(k)X(k))^2 - e^2(k-1)$$

$$= (d(k) - (H^T(k-1) + g^T(k)\alpha(k))X(k))^2 - e^2(k-1)$$

$$= (d(k) - H^T(k-1)X(k) - g^T(k)\alpha(k)X(k))^2 - e^2(k-1)$$

$$= (\alpha(k) - g^T(k)\alpha(k)X(k))^2 - e^2(k-1)$$

$$= \alpha^2(k)\left(1 - g^T(k)X(k)\right)^2 - e^2(k-1) \tag{3.12}$$

Using the expressions (3.8) and (3.9) in the expression (3.12), we have

$$\Delta V(k) = -(1-\kappa^2)e^2(k-1) < 0 \tag{3.13}$$

According to Lyapunov stability theory [20], the error *e(k)* will asymptotically converge to zero.

Remark 3.1: In order to avoid the singularities of the adaptive gain *g(k)* in the expression (3.8) when *//X(k)//* and $\alpha(k)$ approach zero, the following modified adaptive law was proposed:

$$g(k) = \frac{X(k)}{\lambda_1 + \| X(k) \|^2}\left(1 - \kappa \frac{|e(k-1)|}{\lambda_2 + |\alpha(k)|}\right) \tag{3.14}$$

where λ_1, λ_2 are small positive numbers and $0 \leq \kappa < 1$.

Remark 3.2: Section 3.3 has provided only a basic idea of the Lyapunov filtering. Many problems, such as the analysis of convergence rate for the adaptive filtering system in Theorem 3.1 and the convergence region of the adaptive filter using the modified adaptive gain in (3.14) have not been investigated. In the following section we will explore these important properties of the Lyapunov filtering systems in detail.

3.4 Convergence Analysis of Lyapunov Adaptive Filters

The convergence rate of the Lyapunov filters is analyzed in this section. The convergence region of the modified Lyapunov adaptive filter to avoid the singularities is also presented.

Theorem 3.2: Consider the FIR filtering system in (3.2). If the Lyapunov updated law in (3.7) – (3.9) is used to update the filter parameters, the error *e(k)* between the desired reference signal *d(k)* and the filter output *y(k)* can converge to zero exponentially.

Proof: Using (3.3), (3.7) – (3.9), the error *e(k)* can be expressed as

$$
\begin{aligned}
e(k) &= d(k) - y(k) \\
&= d(k) - H^T(k)X(k) \\
&= d(k) - \left[H^T(k-1) + g^T(k)\alpha(k)\right]X(k) \\
&= d(k) - H^T(k-1)X(k) - g^T(k)\alpha(k)X(k) \\
&= \alpha(k) - g^T(k)\alpha(k)X(k) \\
&= \alpha(k) - \frac{X^T(k)}{\| X(k) \|^2}\left(1 - \kappa\frac{|e(k-1)|}{|\alpha(k)|}\right)\alpha(k)X(k) \\
&= \alpha(k) - \left(1 - \kappa\frac{|e(k-1)|}{|\alpha(k)|}\right)\alpha(k) \\
&= \kappa\frac{\alpha(k)}{|\alpha(k)|}|e(k-1)| \\
&= \kappa|e(k-1)|\,\mathrm{sgn}(\alpha(k)) \qquad (3.15)
\end{aligned}
$$

$$
\begin{aligned}
\therefore\ & |e(k)| = \kappa|e(k-1)| \\
& |e(1)| = \kappa|e(0)| \\
& |e(2)| = \kappa|e(1)| = \kappa^2|e(0)| \\
& \vdots
\end{aligned}
$$

$$| e(k) | = \kappa^k | e(0) | \tag{3.16}$$

Remark 3.3: The expression (3.16) shows that the error *e(k)* converges to zero exponentially and the convergence rate is controlled by the positive constant κ. The smaller κ is, the faster the error converges.

Theorem 3.3: Consider the FIR filter system in (3.2). If the filter parameters are updated according to the following modified adaptive law:

$$H(k) = H(k - 1) + g(k)\alpha(k) \tag{3.17}$$

$$g(k) = \frac{X(k)}{\lambda_1 + \| X(k) \|^2}\left(1 - \kappa \frac{| e(k-1) |}{\lambda_2 + | \alpha(k) |}\right) \tag{3.18}$$

$$\alpha(k) = d(k) - H^T(k-1)X(k) \tag{3.19}$$

where λ_1, λ_2 are small positive numbers and $0 \le \kappa < 1$, then the error *e(k)* will converge to the ball at the origin of the error space with the radius

$$r_{e1} = \frac{-\frac{3\kappa\lambda_2\bar{\lambda}}{2} + \sqrt{\left(\frac{3\kappa\lambda_2\bar{\lambda}}{2}\right)^2 + 4\left(1 - \frac{\kappa^2\bar{\lambda}^2}{4}\right)\frac{9}{4}\lambda_2^2}}{-2\left(1 - \frac{\kappa^2\bar{\lambda}^2}{4}\right)} \tag{3.20}$$

where $\bar{\lambda}$ is a constant discussed later.

Proof: Define a Lyapunov function

$$V(k) = e^2(k) \tag{3.21}$$

We then have

$$\Delta V(k) = V(k) - V(k-1)$$

$$
\begin{aligned}
&= \alpha^2(k)\left(1 - g^T(k)X(k)\right)^2 - e^2(k-1) \\
&= \alpha^2(k)\left[1 - \frac{\| X(k)\|^2}{\lambda_1 + \| X(k)\|^2}\left(1 - \kappa\frac{|e(k-1)|}{\lambda_2 + |\alpha(k)|}\right)\right]^2 - e^2(k-1) \\
&= \alpha^2(k)\left[1 - \frac{\frac{\| X(k)\|^2}{\lambda_1}}{1 + \frac{\| X(k)\|^2}{\lambda_1}}\left(1 - \frac{\kappa}{\lambda_2}\frac{|e(k-1)|}{1 + \frac{|\alpha(k)|}{\lambda_2}}\right)\right]^2 - e^2(k-1)
\end{aligned}
\tag{3.22}
$$

where $\frac{\| X(k)\|^2}{\lambda_1} < 1$ and $\frac{|\alpha(k)|}{\lambda_2} < 1$ (3.23)

Then, the following equations are obtained by using Taylor series.

$$
\frac{\frac{\| X(k)\|^2}{\lambda_1}}{1 + \frac{\| X(k)\|^2}{\lambda_1}} = \frac{\| X(k)\|^2}{\lambda_1} + O\left(\frac{\| X(k)\|^2}{\lambda_1}\right) \tag{3.24}
$$

and $$\frac{1}{1 + \frac{|\alpha(k)|}{\lambda_2}} = 1 - \frac{|\alpha(k)|}{\lambda_2} + O\left(\frac{|\alpha(k)|}{\lambda_2}\right) \tag{3.25}$$

where

$$
\left|O\left(\frac{\| X(k)\|^2}{\lambda_1}\right)\right| = \left|\frac{\| X(k)\|^2}{\lambda_1 + \| X(k)\|^2} - \frac{\| X(k)\|^2}{\lambda_1}\right| = \frac{\| X(k)\|^4}{\lambda_1(\lambda_1 + \| X(k)\|^2)}
$$

$$
= \frac{\| X(k)\|^2}{\lambda_1} \cdot \frac{\| X(k)\|^2}{\lambda_1 + \| X(k)\|^2} \le 1 \times \frac{1}{2} = \frac{1}{2} \tag{3.26}
$$

and $$\left|O\left(\frac{|\alpha(k)|}{\lambda_2}\right)\right| = \left|\frac{\lambda_2}{\lambda_2 + |\alpha(k)|} - 1 + \frac{|\alpha(k)|}{\lambda_2}\right| = \frac{|\alpha(k)|^2}{\lambda_2(\lambda_2 + |\alpha(k)|)}$$

$$
= \frac{|\alpha(k)|}{\lambda_2} \frac{|\alpha(k)|}{(\lambda_2 + |\alpha(k)|)} \le 1 \times \frac{1}{2} = \frac{1}{2} \tag{3.27}
$$

Then expression (3.22) can be written as

$$
\begin{aligned}
&\Delta V(k)\\
&= \alpha^2(k)\left[1-\left(\frac{\|X(k)\|^2}{\lambda_1}+O\left(\frac{\|X(k)\|^2}{\lambda_1}\right)\right)\left(1-\frac{\kappa|e(k-1)|}{\lambda_2}\left(1-\frac{|\alpha(k)|}{\lambda_2}+O\left(\frac{|\alpha(k)|}{\lambda_2}\right)\right)\right)\right]^2\\
&\quad - e^2(k-1)\\
&= \alpha^2(k)\left[1-\left(\frac{\|X(k)\|^2}{\lambda_1}+O\left(\frac{\|X(k)\|^2}{\lambda_1}\right)\right)\left(1-\frac{\kappa|e(k-1)|}{\lambda_2}+\frac{\kappa|e(k-1)||\alpha(k)|}{{\lambda_2}^2}\right.\right.\\
&\quad \left.\left.-\frac{\kappa|e(k-1)|}{\lambda_2}O\left(\frac{|\alpha(k)|}{\lambda_2}\right)\right)\right]^2 - e^2(k-1)
\end{aligned}
$$

$$
\begin{aligned}
&= \alpha^2(k)\left[1-\frac{\|X(k)\|^2}{\lambda_1}+\frac{\kappa\|X(k)\|^2|e(k-1)|}{\lambda_1\lambda_2}-\frac{\kappa\|X(k)\|^2|e(k-1)||\alpha(k)|}{\lambda_1{\lambda_2}^2}\right.\\
&+\frac{\kappa\|X(k)\|^2|e(k-1)|}{\lambda_1\lambda_2}O\left(\frac{|\alpha(k)|}{\lambda_2}\right)-O\left(\frac{\|X(k)\|^2}{\lambda_1}\right)+\frac{\kappa|e(k-1)|}{\lambda_2}O\left(\frac{\|X(k)\|^2}{\lambda_1}\right)\\
&\left.-\frac{\kappa|e(k-1)||\alpha(k)|}{{\lambda_2}^2}O\left(\frac{\|X(k)\|^2}{\lambda_1}\right)+\frac{\kappa|e(k-1)|}{\lambda_2}O\left(\frac{\|X(k)\|^2}{\lambda_1}\right)O\left(\frac{|\alpha(k)|}{\lambda_2}\right)\right]^2\\
&- e^2(k-1)
\end{aligned}
$$

$$
\begin{aligned}
&\le \alpha^2(k)\left[1+\frac{\kappa\|X(k)\|^2|e(k-1)|}{\lambda_1\lambda_2}+\frac{\kappa\|X(k)\|^2|e(k-1)|}{\lambda_1\lambda_2}O\left(\frac{|\alpha(k)|}{\lambda_2}\right)\right.\\
&-O\left(\frac{\|X(k)\|^2}{\lambda_1}\right)+\frac{\kappa|e(k-1)|}{\lambda_2}O\left(\frac{\|X(k)\|^2}{\lambda_1}\right)-\frac{\kappa|e(k-1)||\alpha(k)|}{{\lambda_2}^2}O\left(\frac{\|X(k)\|^2}{\lambda_1}\right)\\
&\left.+\frac{\kappa|e(k-1)|}{\lambda_2}O\left(\frac{\|X(k)\|^2}{\lambda_1}\right)O\left(\frac{|\alpha(k)|}{\lambda_2}\right)\right]^2 - e^2(k-1)
\end{aligned}
\qquad (3.28)
$$

It is noted that

$$
\begin{aligned}
\Delta V(k) &< {\lambda_2}^2\left[1+\frac{\kappa|e(k-1)|}{\lambda_2}+\frac{\kappa|e(k-1)|}{2\lambda_2}+\frac{1}{2}+\frac{\kappa|e(k-1)|}{2\lambda_2}\right.\\
&\quad \left.+\frac{\kappa\lambda_1|e(k-1)|}{2\lambda_2}+\frac{\kappa|e(k-1)|}{4\lambda_2}\right]^2 - e^2(k-1)\\
&= {\lambda_2}^2\left[\frac{3}{2}+\frac{\kappa|e(k-1)|}{2\lambda_2}(2+1+1+\lambda_1+\frac{1}{2})\right]^2 - e^2(k-1)\\
&= {\lambda_2}^2\left[\frac{3}{2}+\frac{\kappa|e(k-1)|}{2\lambda_2}(4\frac{1}{2}+\lambda_1)\right]^2 - e^2(k-1)
\end{aligned}
\qquad (3.29)
$$

and Let $\bar{\lambda} = 4\frac{1}{2} + \lambda_1$ (3.30)

$$\begin{aligned}\Delta V(k) &< \lambda_2^2\left[\frac{3}{2} + \frac{\kappa\bar{\lambda}}{2\lambda_2}|e(k-1)|\right]^2 - e^2(k-1) \\ &= \lambda_2^2\left[\frac{9}{4} + \frac{3\kappa\bar{\lambda}}{2\lambda_2}|e(k-1)| + \frac{\kappa^2\bar{\lambda}^2}{4\lambda_2^2}|e^2(k-1)|\right]^2 - e^2(k-1) \\ &= -\left(1 - \frac{\kappa^2\bar{\lambda}^2}{4}\right)e^2(k-1) + \frac{3\kappa\lambda_2\bar{\lambda}}{2}|e(k-1)| + \frac{9}{4}\lambda_2^2\end{aligned}$$

(3.31)

For the further analysis, we consider the following parabolic function:

$$\Delta\bar{V}(k) = -\left(1 - \frac{\kappa^2\bar{\lambda}^2}{4}\right)e^2(k-1) + \frac{3\kappa\lambda_2\bar{\lambda}}{2}|e(k-1)| + \frac{9}{4}\lambda_2^2 \quad (3.32)$$

If κ is small enough in the sense that,

$$\frac{\kappa^2\bar{\lambda}^2}{4} < 1 \quad \text{or} \quad \frac{\kappa\bar{\lambda}}{2} < 1 \quad (3.33)$$

and $$\kappa\left(\frac{9}{2} + \lambda_1\right) < 2 \quad (3.34)$$

$\Delta\bar{V}(k)$ in the expression (3.32) is a concave down parabolic function.

Also, for the given λ_1 and λ_2, the small positive number κ satisfies the following inequality

$$0 \le \kappa < 2/\left(\frac{9}{2} + \lambda_1\right) \quad (3.35)$$

Solving the quadratic equation $\Delta\bar{V}(k) = 0$, we obtain the two roots as follow:

$$r_{e\,1,2} = \frac{-\frac{3\kappa\lambda_2\bar{\lambda}}{2} \pm \sqrt{\left(\frac{3\kappa\lambda_2\bar{\lambda}}{2}\right)^2 + 4\left(1 - \frac{\kappa^2\bar{\lambda}^2}{4}\right)\frac{9}{4}\lambda_2^2}}{-2\left(1 - \frac{\kappa^2\bar{\lambda}^2}{4}\right)} \quad (3.36)$$

The root r_{el} is considered because $|e(k-1)| \ge 0$

$$r_{e1} = \frac{-\frac{3\kappa\lambda_2\overline{\lambda}}{2} + \sqrt{\left(\frac{3\kappa\lambda_2\overline{\lambda}}{2}\right)^2 + 4\left(1-\frac{\kappa^2\overline{\lambda}^2}{4}\right)\frac{9}{4}\lambda_2^2}}{-2\left(1-\frac{\kappa^2\overline{\lambda}^2}{4}\right)} \tag{3.37}$$

Therefore, the error $|e(k-1)|$ should satisfy the following equality

$$|e(k-1)| > r_{e1} \tag{3.38}$$

in order to make $\Delta V(k) < \Delta\overline{V}(k) < 0$. Then the error will converge to the ball center at the origin of the error space with radius r_{e1} in the expression (3.20).

Remark 3.4: It is seen, from the introduction and the analysis of the error convergence properties in the above, that only the input and output measurements are used for the design of the Lyapunov adaptive filters. Hence the stochastic properties of the signals do not affect the performance of the filters. The main reason is that the optimization technique used here is based on the Lyapunov stability theory and is not based on the gradient search techniques. It is known that the gradient search–based optimization is indeed affected by the stochastic properties of the signals. In our approach, if the input disturbances are bounded random processes, the adaptive filtering algorithm can be directly designed using the input and output measurements based on the Lyapunov stability theory without considering the stochastic properties of the signals. This point is similar to the design of Lyapunov stability based adaptive control systems and variable structure control systems [20].

Remark 3.5: The Lyapunov stability theory used for the design of adaptive filters in this chapter provides an optimization method in state space. However, it is different from the gradient search based methods. According to Lyapunov stability theory, the selected $V(k)$ is a Lyapunov function if and only is $\Delta V(k)$ is negative ($\Delta V(k) < 0$). For the Lyapunov adaptive filtering system in this chapter, whether or not $\Delta V(k)$ is negative depends on the selection of the parameter updated law. Only when the parameter update law of the filtering system is chosen in Lyapunov sense, is $V(k)$ a Lyapunov function of the designed adaptive filtering system, which has a unique global minimum. Therefore, the selection of the Lyapunov function and the parameter updated law are not independent, the proper selection of the parameter

updated law can guarantee that function $V(k)$ is a Lyapunov function of the adaptive filtering system with a unique global minimum in the state space.

On the other hand, the cost function of a gradient search-based adaptive filtering system has a fixed structure in the parameter space after the expression of the cost function is chosen. The parameter update law is only a means to search for the global minimum. The parameter update law is independent of the cost function in the parameter space.

Remark 3.6: Although the cost function and the Lyapunov function have many different characteristics, they are all energy-like functions. One is considered in the state space, and another in the parameter space. The corresponding optimization methods can be used to design adaptive filters with different requirements.

3.5 Computational Complexity Analysis of LAF

Computational complexity is another essential quantity to measure the effectiveness of an adaptive algorithm. The LMS algorithms have a complexity which is typically close to L multiplies per weight vector update (L is the filter order). RLS techniques have a complexity that is proportional to L^2. The complexity of the presented LAF is analysed as follow:

	Multiplies	**Divides**	**Adds**	**Substrates**
$\alpha(k)= d(k) - H^T(k-1)X(k)$	L			1
$g(k) = \frac{X(k)}{\| X(k) \|^2}\left(1-\kappa\frac{\| e(k-1) \|}{\| \alpha(k) \|}\right)$	2L+1	L+1		1
or				
$g(k) = \frac{X(k)}{\lambda_1+\| X(k) \|^2}\left(1-\kappa\frac{\| e(k-1) \|}{\lambda_2+\| \alpha(k) \|}\right)$	2L+1	L+1	2	1
$H(k) = H(k - 1) + g(k)\alpha(k)$	L		L	
$e(k) = d(k) - H^T(k)X(k)$	L			1

From the above analysis, the LAF algorithm has the computational complexity that is about $5L$ multiplies per weight vector update ($\propto L$). Hence the computational requirement of LAF ($\propto L$) is lower than that of RLS ($\propto L^2$). For large filter orders, the adaptive algorithms that have higher computational complexity can give difficulty in real-time implementation. Therefore, computational complexity is an important quantity to measure the effectiveness of an adaptive algorithm.

3.6 Lyapunov Theory-based Adaptive Filtering (LAF) for IIR Filter

Over the last several years, adaptive *infinite impulse response* (IIR) filtering has been an active area of research and it has been considered for a variety of problems in signal processing and communication. An adaptive IIR filter can provide significantly better performance than an adaptive FIR filter having the same number of coefficients. This is a consequence of the output feedback that generates an infinite impulse response with only a finite number of parameters. A desired response or, equivalently, its frequency response can be approximated more effectively by the output of a filter that has both poles and zeros compared to FIR that has only zeros. For example, an IIR filter with sufficient order can exactly model an unknown pole-zero system, whereas the FIR filter can only approximate such a system. Alternatively, to achieve a specified level of performance, an IIR filter generally requires considerably fewer coefficients than the corresponding FIR filter. But this has to be offset by the fact that the stability of an IIR filter can no longer be guaranteed. Furthermore, this also requires the expense of increased estimation complexity involving RLS type estimators [9],[12],[16],[17]. A full discussion of these properties, including methods for monitoring the parameter updates and resetting any unstable effects, is available in the literature [1].

Basically, there are two approaches to adaptive IIR filtering that correspond to different formulations of the error. These are known as *equation error* and *output error* methods [18]. One of drawbacks of the *output error* IIR method is that unlike the adaptive FIR, the performance surface is not hyper-paraboloid and has local minima to which the algorithm can converge. If the performance criterion is based on

the *equation error* formulation then the performance surface is a hyper-paraboloid, but this approach can lead to the global minimum is biased away from the optimal solution in the presence of noise and produce erroneous solutions [18]. Another disadvantage of any adaptive IIR system is that if the poles move outside the unit circle, the IIR filter itself will become unstable in addition to the normal stability concerns for the adaptive algorithms such as the step size for gradient based adaptive algorithms. Although different approaches have been suggested to monitor the system poles or to use a different structure, e.g. parallel structures [21], cascade structures [22] or lattice structures [23],[24], additional computational complexity is introduced.

The IIR filter can implemented using a pole-zero structure or ARMA-output error (*autoregressive moving average*) model that allows regressed input output term. It can be characterized by the difference equation (3.39)

$$y(k) = \sum_{i=0}^{N-1} b_i(k)x(k-i) + \sum_{i=1}^{N-1} a_i(k)y(k-i) \tag{3.39}$$

The recursive difference equation in (3.39) can be rewritten as

$$y(k) = B^T(k)\tilde{X}(k) + A^T(k)\tilde{Y}(k-1) = H^T(k)X(k) \tag{3.40}$$

where $$A(k) = [a_1(k), a_2(k), \cdots, a_{N-1}(k)]^T \tag{3.41}$$

$$B(k) = [b_0(k), b_1(k), \cdots, b_{N-1}(k)]^T \tag{3.42}$$

$$\tilde{X}(k) = [x(k), x(k-1), \cdots, x(k-N+1)]^T \tag{3.43}$$

$$\tilde{Y}(k) = [y(k-1), y(k-2), \cdots, y(k-N+1)]^T \tag{3.44}$$

$$H(k) = [b_0(k), b_1(k), \cdots, b_{N-1}(k), a_1(k), a_2(k), \cdots, a_{N-1}(k)]^T \tag{3.45}$$

$$X(k) = [x(k), x(k-1), \cdots, x(k-N+1), y(k-1), y(k-2), \cdots, y(k-N+1)]^T \tag{3.46}$$

Remark 3.7: The design principle of the Lyapunov filtering given in theorem 3.1 can also be implemented if $y(k)$, $H(k)$ and $X(k)$ are specified by (3.40), (3.44) and (3.45). It is easy to prove that, for IIR filtering system in (3.40), the error can also exponentially converge to zero if the adaptive law in (3.7)–(3.9) is used with specified $y(k)$, $H(k)$, and $X(k)$ in (3.40), (3.44) and (3.45), respectively.

Remark 3.8: The stability of the error dynamics between the desired reference signal $d(k)$ and the filter output can be guaranteed based on the Lyapunov stability theory [20]. This indicates that the normal stability concerns for the adaptive algorithm is guaranteed by the Lyapunov Stability Theory. However, the stability of the overall system is only assured if the unknown system itself is stable if operating in system identification.

Remark 3.9: Adaptive algorithms based on the *output-error* formulation are generally more complicated than those based on the *equation error*, but they do not lead to biased solutions. However, they may converge to a local minimum of the MSOE (*mean square output error*) surface [18] because the MSOE surface generally is not a paraboloid and it can have local minima. In addition, the initial conditions for $X(k)$ can also influence to which minimum the algorithm will converge. It is obviously desirable that $X(0)$ is near or lies on a trajectory to the global minimum. Design of an adaptive IIR filter using Lyapunov theory provides a viable solution to the sub-optimal problem. As mentioned before, unlike the gradient-descent method that searches for the minimum of MSOE, Lyapunov method does not perform searching in the error surface. The performance criterion is the Lyapunov function, $V(k)=e^2$ which is a quadratic function with a single global minimum. According to Lyapunov theory [20], if a positive define function, $V(k)=e^2$ is found such that its discrete time difference taken along a trajectory is always negative $\Delta V(k)<0$, then as time k increases, $V(k)$ will finally converge to zero and therefore the error will also converge to zero asymptotically. Again, whether or not $\Delta V(k)$ is negative depends on the selection of the parameter update law. Only when the parameter update law of the filtering system is chosen in Lyapunov sense, is $V(k)$ a Lyapunov function of the designed adaptive filtering system, which has a unique global minimum. Therefore, the selection of the Lyapunov function and the parameter update law are not independent, the proper selection of the parameter update law can guarantee that

function $V(k)$ is a Lyapunov function of the adaptive filtering system with a unique global minimum in the state space. Hence, this approach is not sensitive to initial conditions for global minimum convergence in the IIR structure.

Remark 3.10: Most existing adaptive IIR algorithms are based on gradient search methods such as the Gauss-Newton [3],[6] and implemented in the *output-error* and *equation error* formulations. The convergence rate of Gauss-Newton depends on the *step-size*. In order for it to converge, the Hessian matrix must always be positive definite (invertable) and the system poles must always lie inside the unit circle. In addition, this algorithm is computationally expensive due to the Hessian matrix updated. *Recursive prediction error* (PRE) algorithm [1],[18],[25] is a gradient-descent approach. It adjusts the filter coefficients to minimize the MSOE cost function. However, it requires significant amount of complexity and large amount of storage. An approximation to the gradient leads to simpler algorithm known as *pseudolinear regression* (PLR) algorithm [18],[25]. It is similar in form to RLS widely used in *equation-error* formulation. The computation complexity and storage requirements are comparable to that of RLS, and they are clearly less than that of RPE. Designing the adaptive IIR filter using Lyapunov theory offers an alternative approach to the stability of IIR filter. The Lyapunov IIR filter has computational complexity that is less than that of most exiting adaptive IIR algorithms such as Gaussion-Newton, RPE, PLR, RLS, and IIR-QR. The overall complexity for IIR-QR is $O(2N^2)$, where N is the number of the total inputs the filter. The computational requirements for a simplified gradient RPE is $O(5N^2)$. The computational complexity of PLR is comparable to that of RLS, $O(4N^2)$, and they are clearly less than that of the RPE algorithm. Thus the Lyapunov IIR filter, with computational complexity $O(5N)$ approximately, has less computation complexity compared to the above algorithms. Table 3.1 give a comparison of the cost per iteration of the aforementioned adaptive algorithms for IIR filter.

Table 3.1	
Adaptive Algorithm	**Cost per iteration**
IIR-QR	$O(\propto N^2)$
Simplified gradient RPE	$O(\propto N^2)$
PLR	$O(\propto N^2)$
RLS	$O(\propto N^2)$
Lyapunov IIR	$O(\propto N)$

Remark 3.11: There have been relatively few analyses of the convergence properties of adaptive IIR filters. Most of the results are derived from work in system identification where it is often assumed that the *step-size* in Gauss-Newton or its modifications decrease to zero with time. That is the adaptive algorithm eventually shuts off. These results are particularly useful for the system identification application where the unknown system is time-invariant and the signal is stationary. In this case, the Gauss-Newton algorithm will converge to a stable point of the ODE (ordinary differential equation) with *probability one* provided the data is asymptotically mean stationary and exponentially stable [18]. For time-varying and non-stationary signals, these algorithms converge only in *probability*. Furthermore, ODE does not prove convergence to the global minimum and provide any information concerning rate of convergence. In the Lyapunov IIR filter, the error convergence is guaranteed. The convergence analysis properties is similar to that in the section 3.4.

3.7 Simulation Examples

In this section, four simulation examples that illustrate the performance of the *Lyapunov Theory-based adaptive filtering* (LAF) adaptive FIR and IIR filters. The first example demonstrates the performance of the Lyapunov FIR design when an additive noise is introduced at the filter input. Simulation of the same setup with RLS is also accomplished for comparison. The second simulation considers the round off error that can cause the unstable behavior of adaptive algorithm is also presented to show the robustness to round off error of this scheme compared to RLS. The third and fourth examples illustrate the use of Lyapunov IIR filter for nonlinear system identification.

Example 1: *Adaptive filtering with Lyapunov FIR filter – additive noise*

In this example, the filter input signal shown in *Figure 3.1* is corrupted with the noise *n(k)*, where *n(k)* is a bounded random noise which satisfies the following bounded condition $0 \leq n(k) < 0.4$. The adaptive gain is updated according to the expression (3.8). In the first case, the parameters λ_1, λ_2 and κ in the expression (3.8) are chosen as follow: $\lambda_1 = \lambda_2 = 0.4$, and $\kappa = 0.8$. The result illustrated in *Figure 3.2* shows the comparison of the reference signal *d(k)* and the filter output signal *y(k)*. It is seen that, although the output of the adaptive filter can follow the desired reference signal well, but the effects of the noise are not fully eliminated because the adaptation rate is relatively slow ($\kappa = 0.8$) and the values of the parameters λ_1, λ_2 are very large.

In the second case, $\lambda_1 = \lambda_2 = 0.01$, and $\kappa = 0.1$ are chosen. *Figure 3.3* shows the comparison of the reference signal *d(k)* and the filter output signal *y(k)*. It can be seen that the effects of the input disturbance has been greatly reduced and the tracking performance between the desired reference signal *d(k)* and the output *y(k)* of the adaptive filter has been greatly improved by properly choosing parameters λ_1, λ_2 and κ. Smaller constant κ also provides faster error convergence. This result has verified the statement in *Theorem 3.2* in the section 3.4. The square output error, $e^2(k)$ of this simulation is displayed in *Figure 3.4*.

For a comparison study, simulation of a third order adaptive filter with RLS algorithm is also presented. The results in *Figure 3.5* (forgetting or weighting factor, ρ= 0.2) reveal the output signal of RLS method has higher noise level compared to that of LAF by observing the square output error, $e^2(k)$ in *Figure 3.6*. This is because LAF has fast convergence speed, good tracking property and is highly stable. The RLS with larger forgetting factor, ρ = 0.5 gives worse output signal but the amplitude variation of the adaptive parameters has become small.

Example 2: *Adaptive filtering with Lyapunov FIR filter – round off error & Large disturbance*

Round off error can affect the performance of the filter. A comparison study of the round off effect for LAF and RLS is also presented. The filter coefficients are rounded off to 2 decimal and this should seriously affect the filter performance. *Figure 3.7* illustrates the corrupted input signal, $x(k)$. The bounded noise is analogous to previous example except a sudden large disturbance is introduced within iterations 500-700. *Figure 3.8* and *Figure 3.9* reveal the output signal, $y(k)$ and the square output error, $e^2(k)$ of LAF-FIR respectively. Simulation results of RLS with forgetting factor and ρ=0.2 are depicted in *Figure 3.10* and *Figure 3.11*. From these results, LAF with FIR design can tolerate the round off error and sudden disturbance and give better performance compared to RLS.

Example 3: *Adaptive filtering with Lyapunov FIR filter - Nonlinear System Identification 1*

To illustrate the performance of adaptive Lyapunov IIR filter, simulations are carried out for nonlinear system identification. A nonlinear SISO system is considered. For a comparison study, simulation with the IIR-RLS algorithm which has the generic form of recursive Gauss-Newton algorithm is also performed. The input signal is white noise with zero mean value and variance 1.

$$y(k) = 0.0705\,x(k) - 0.141\,x(k-1) - 0.0705\,x(k-1) + 1.1993\,e^{-y^2(k-1)}\,y(k-1) - 0.5156\,e^{-y^2(k-2)}\,y(k-2)$$

The simulation results of Lyapunov IIR and IIR-RLS are shown in *Figures 3.12-3.14* and *Figures 3.15-3.17* respectively. A smooth convergence of the parameter $a_i(k)$ is achieved by the Lyapunov IIR filter. These results indicate Lyapunov IIR has better adaptation in the nonlinear system identification.

Example 4: *Adaptive filtering with Lyapunov IIR filter - Nonlinear System Identification 2*

The fourth simulation tests the adaptation performance of these filters when the system poles move temporary outside the unit circle. The unknown transfer function has zeros at 2.4142, -0.4142 and poles at 0.1± 0.6245j inside the unit circle. Within iteration 1000-1500, those poles move outside the unit circle to a new location 1±1.7321j and move back to unit circle. Simulation results with Lyapunov IIR and IIR-RLS are illustrated in *Figure 3.18* and *Figure 3.19*. By observing the period for $b_i(k)$ of both filters to converge back to original values, Lyapunov IIR filter has high adaptation and tracking properties compared to the IIR-RLS.

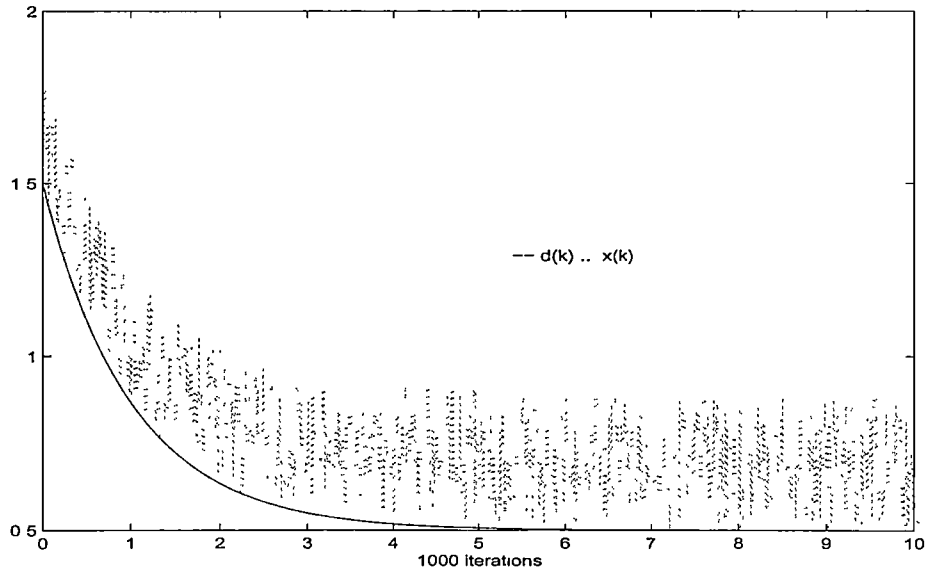


Figure 3.1: The desired response, *d(k)* & the corrupted input signal, *x(k)*

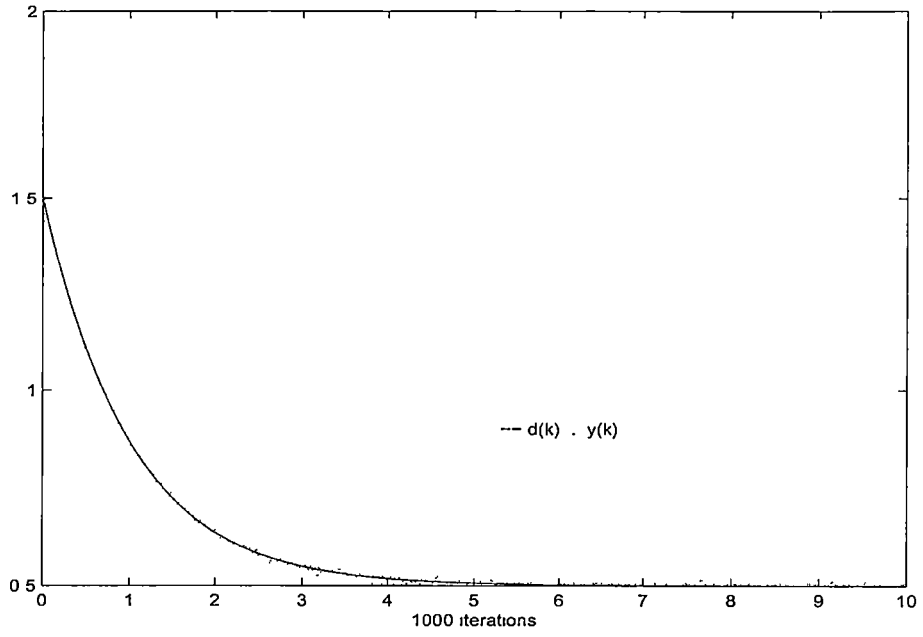


Figure 3.2: LAF-FIR, The desired response, *d(k)* & filter output, *y(k)*

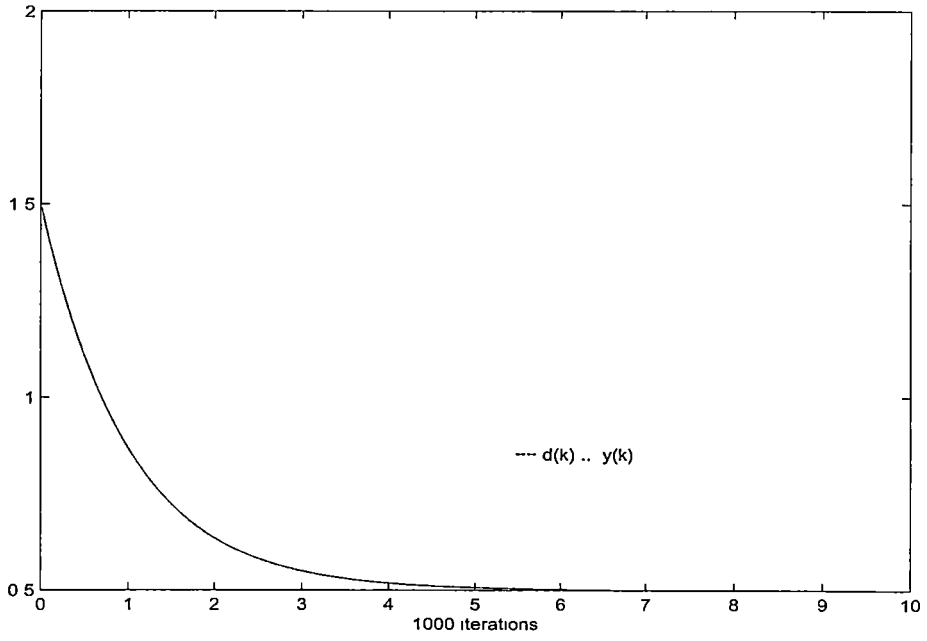


Figure 3.3: LAF-FIR, the desired response, *d(k)* & filter output, *y(k)*

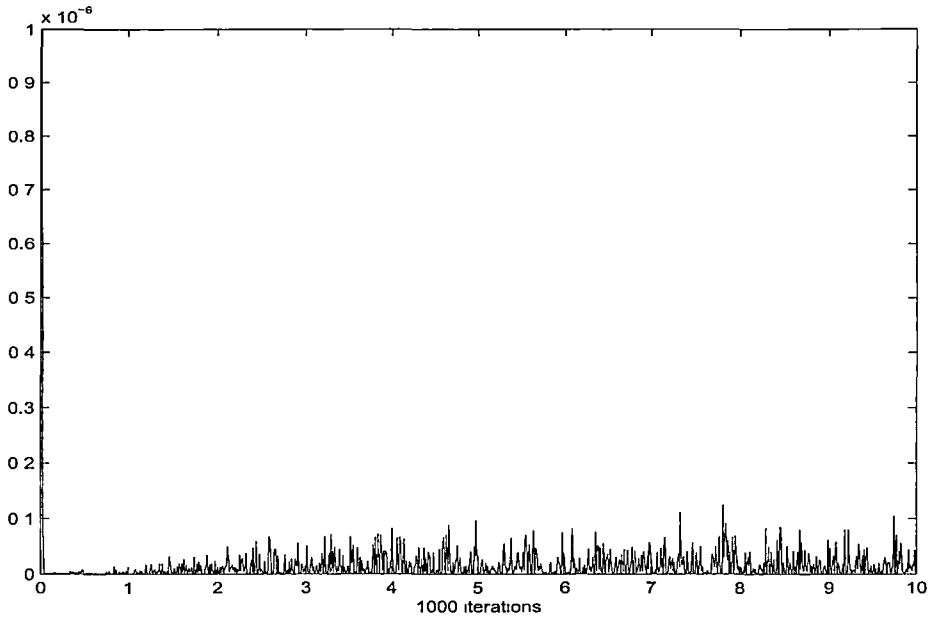


Figure 3.4: LAF-FIR, the square output error, $e^2(k)$

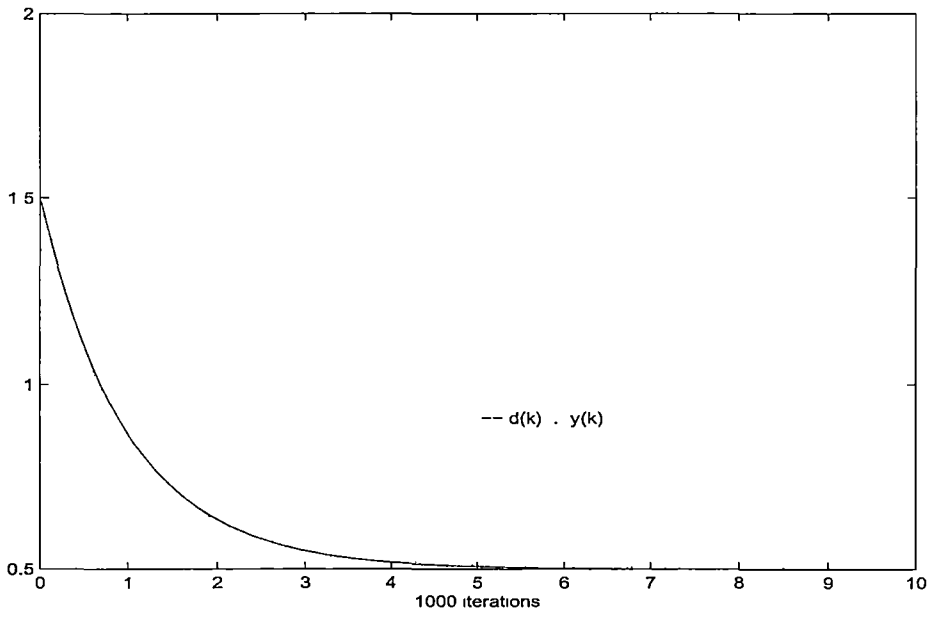


Figure 3.5: RLS (FIR), the desired response, $d(k)$ & filter output, $y(k)$

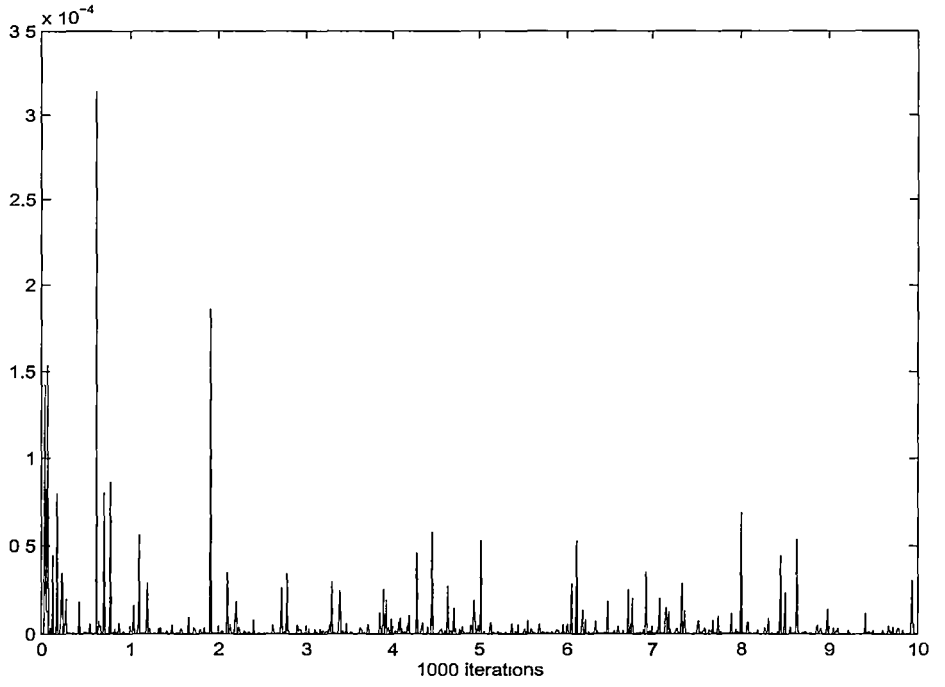


Figure 3.6: RLS (FIR),the square output error, $e^2(k)$

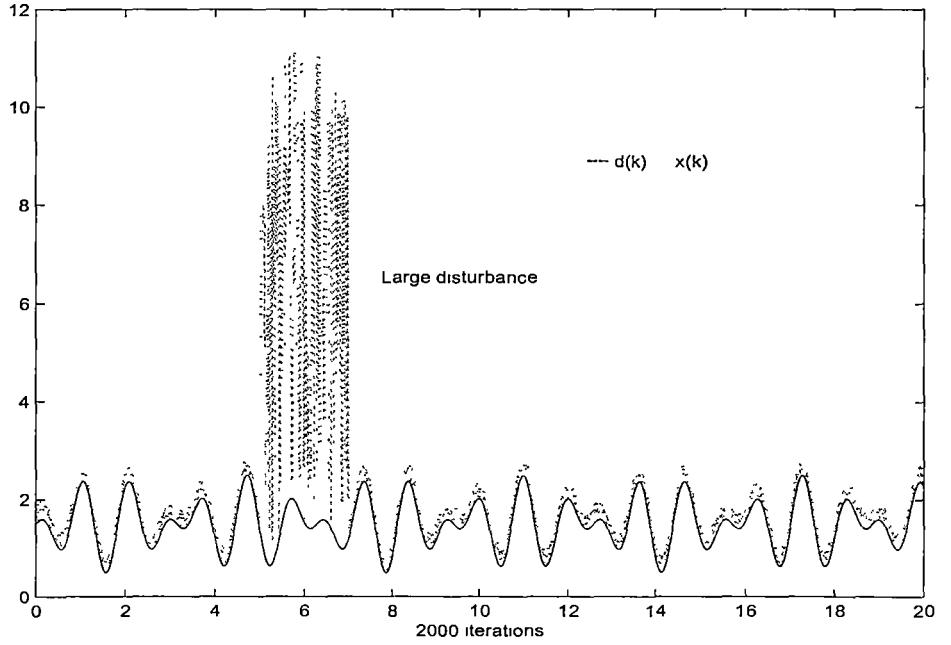


Figure 3.7: The desired response, $d(k)$ & the corrupted input signal, $x(k)$

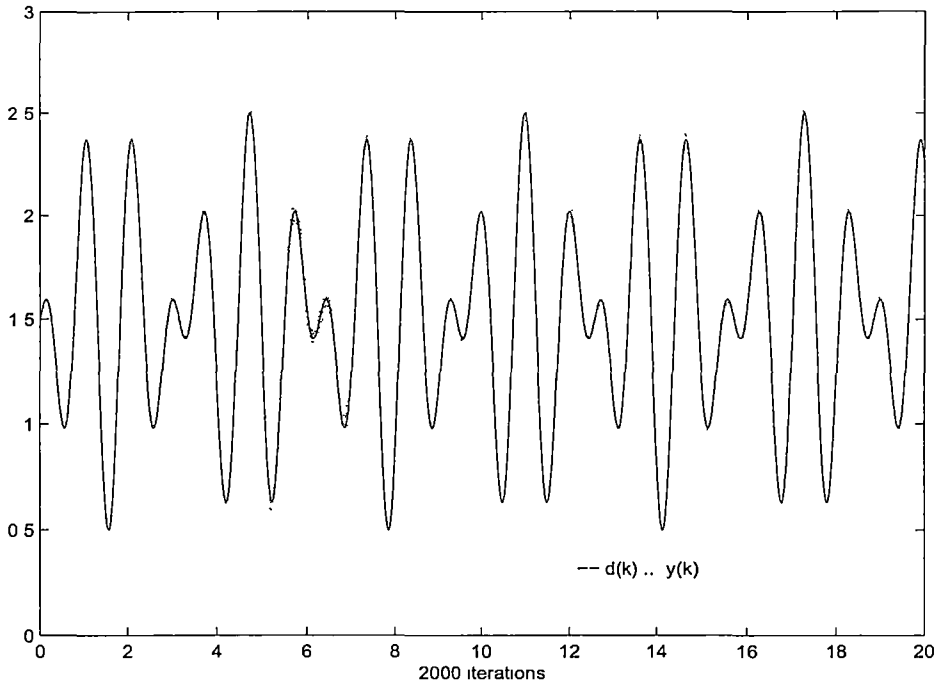


Figure 3.8: LAF-FIR, the desired response, $d(k)$ & filter output, $y(k)$

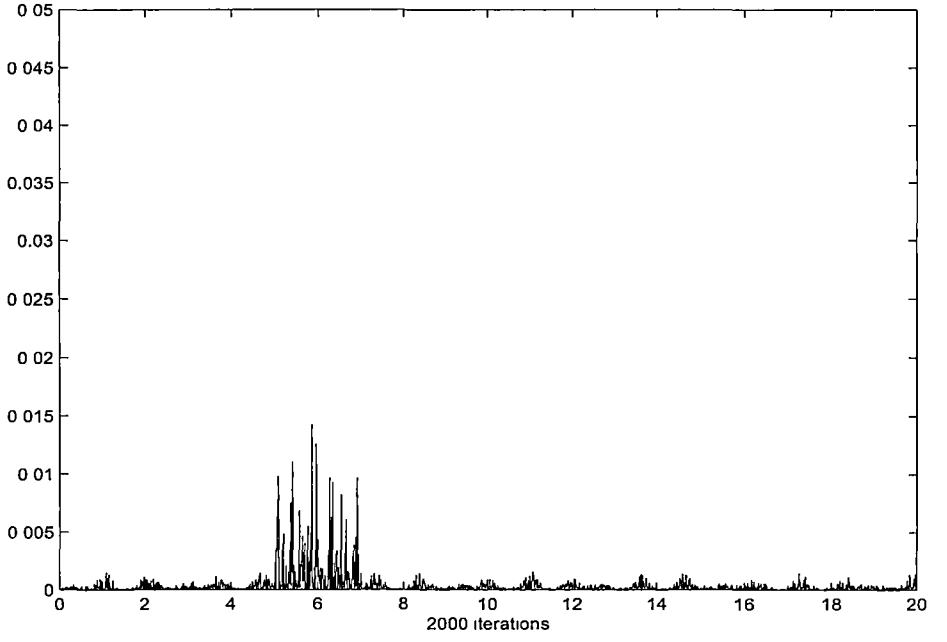


Figure 3.9: LAF-FIR, the square output error, $e^2(k)$

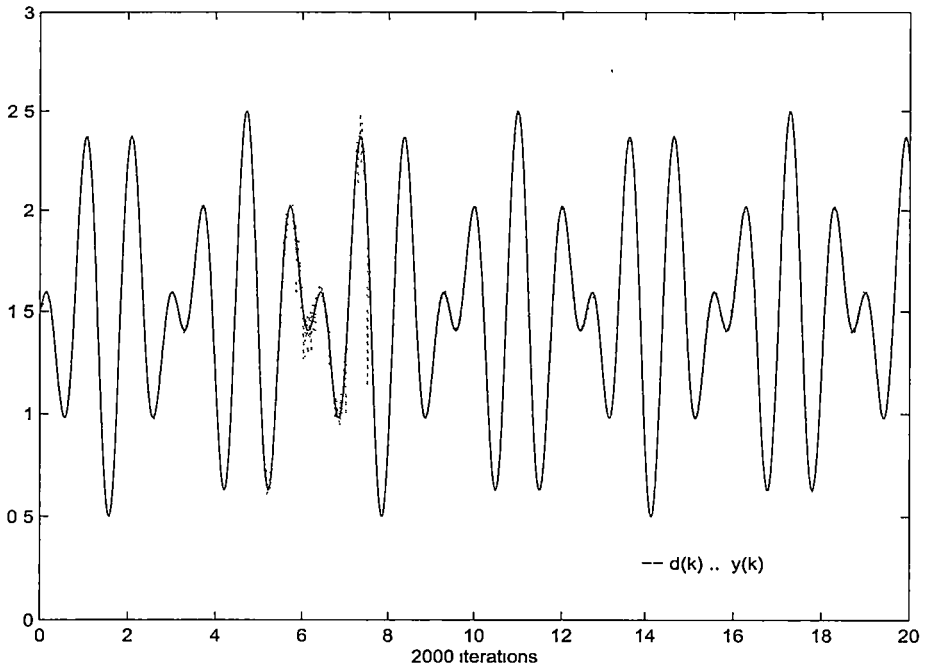


Figure 3.10: RLS (FIR), the desired response, *d(k)* & filter output, *y(k)*

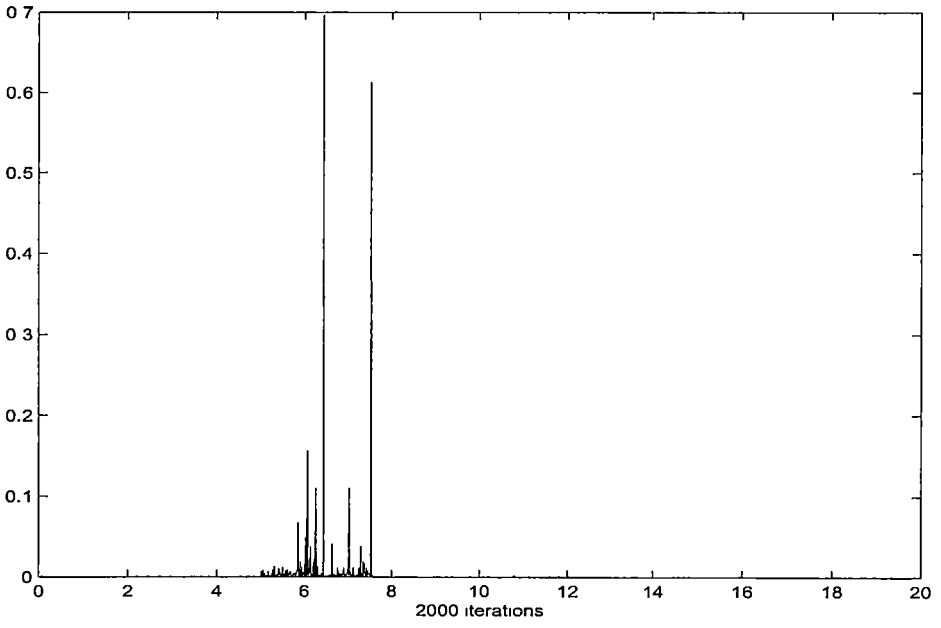
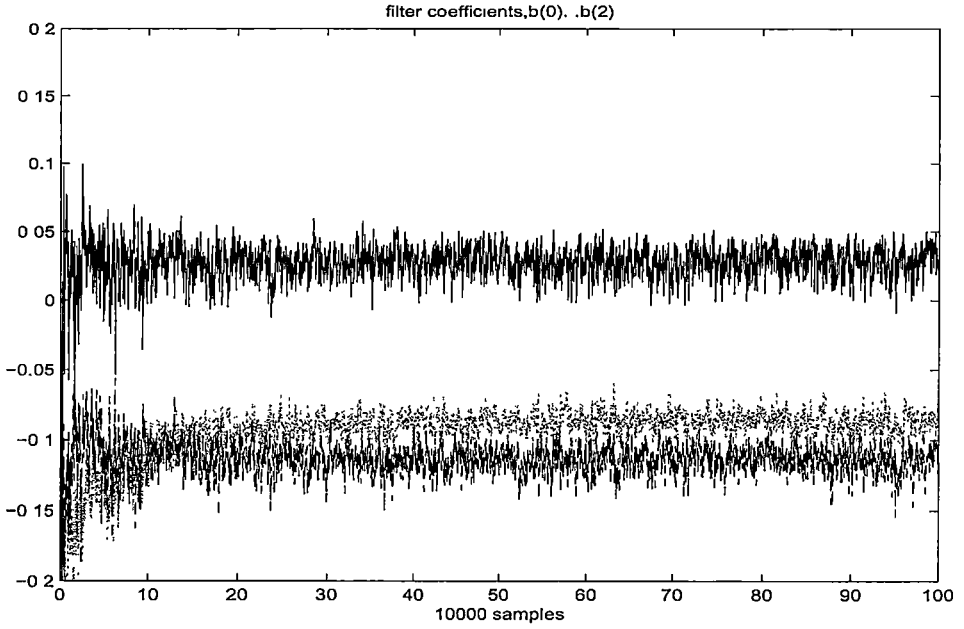


Figure 3.11: RLS (FIR), the square output error, $e^2(k)$


filter coefficients,b(0). .b(2)
0 2
0 15
0.1
0 05
0
−0.05
−0 1
−0 15
−0 2
0
10
20
30
40
50
60
70
80
90
100
10000 samples

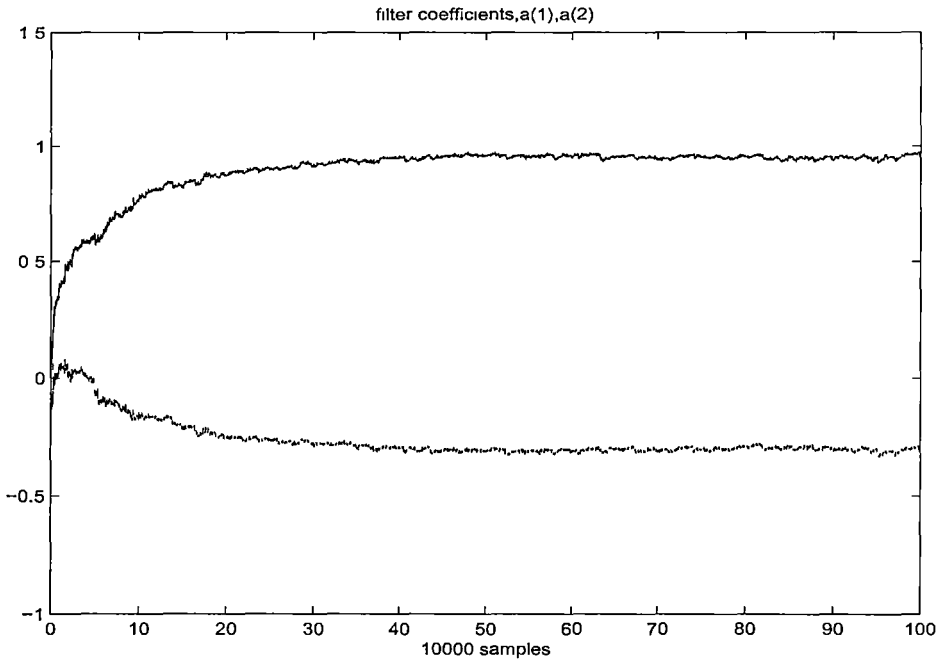


Figure 3.13: LAF-IIR, Feedback coefficients, $a_i(k)$

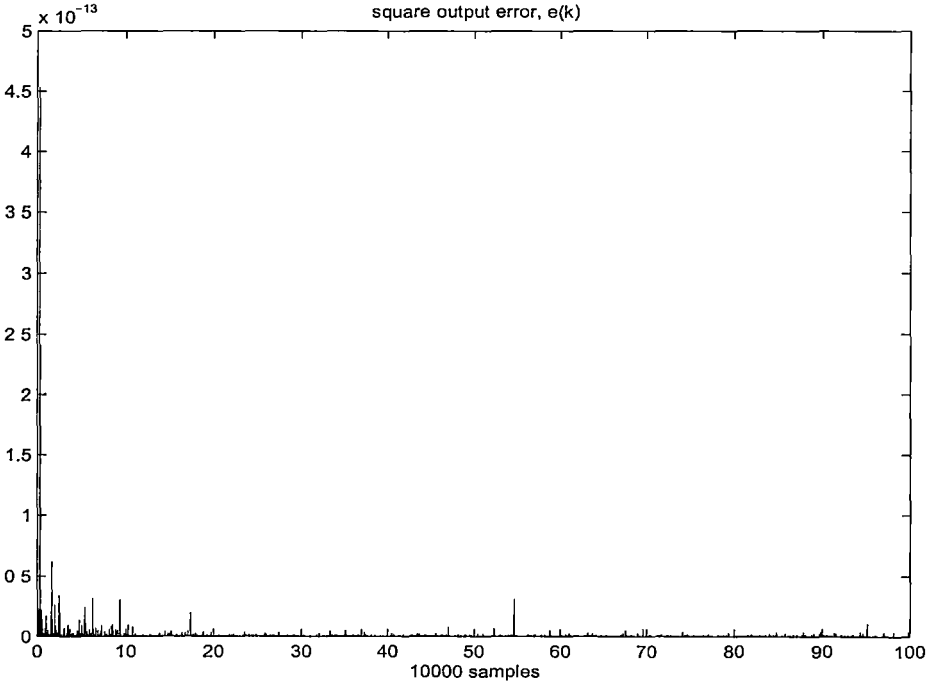


Figure 3.14: LAF-IIR, the square output error, $e^2(k)$ - *y axis* x10^{-13}

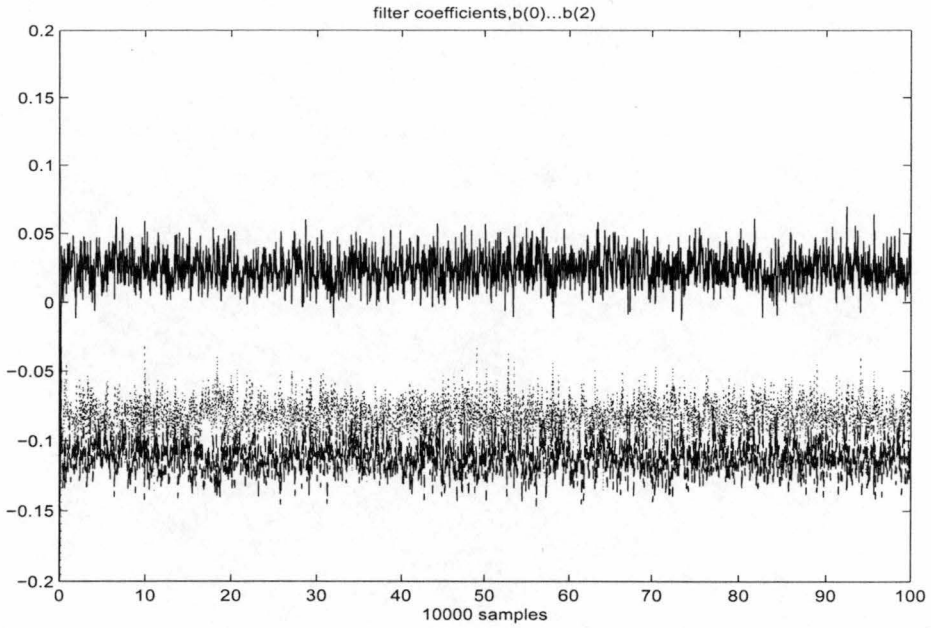


Figure 3.15: RLS (IIR), Forward coefficients, $b_i(k)$

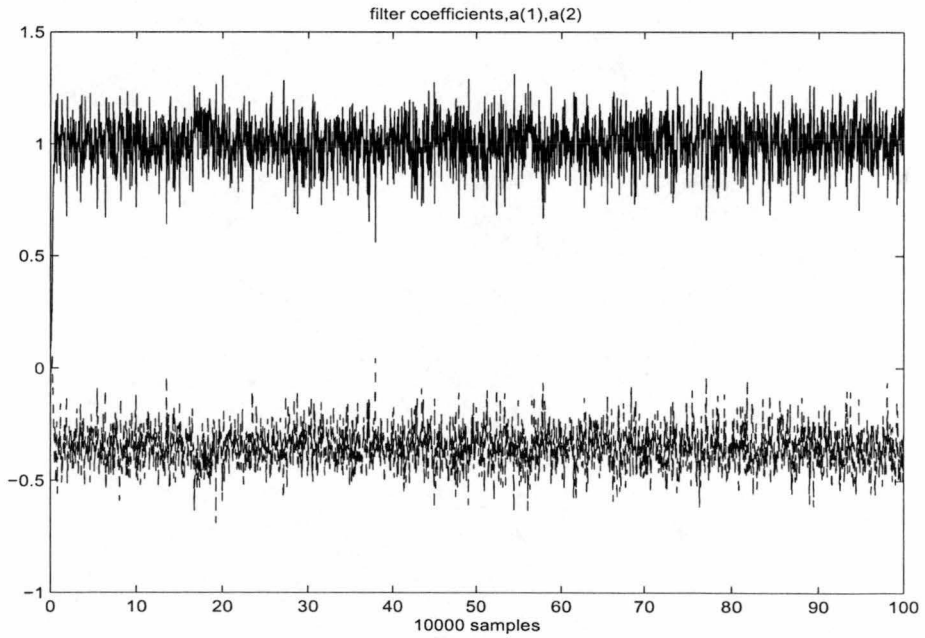


Figure 3.16: RLS (IIR), Feedback coefficients, $a_i(k)$

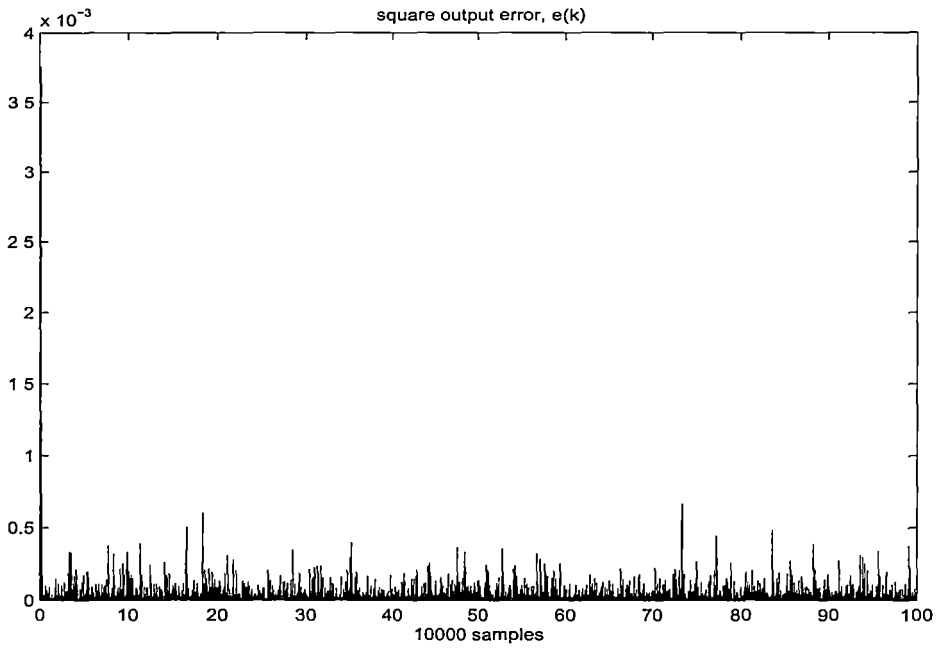


Figure 3.17: RLS (IIR), the square output error, $e^2(k)$ - *y axis* x10^{-3}

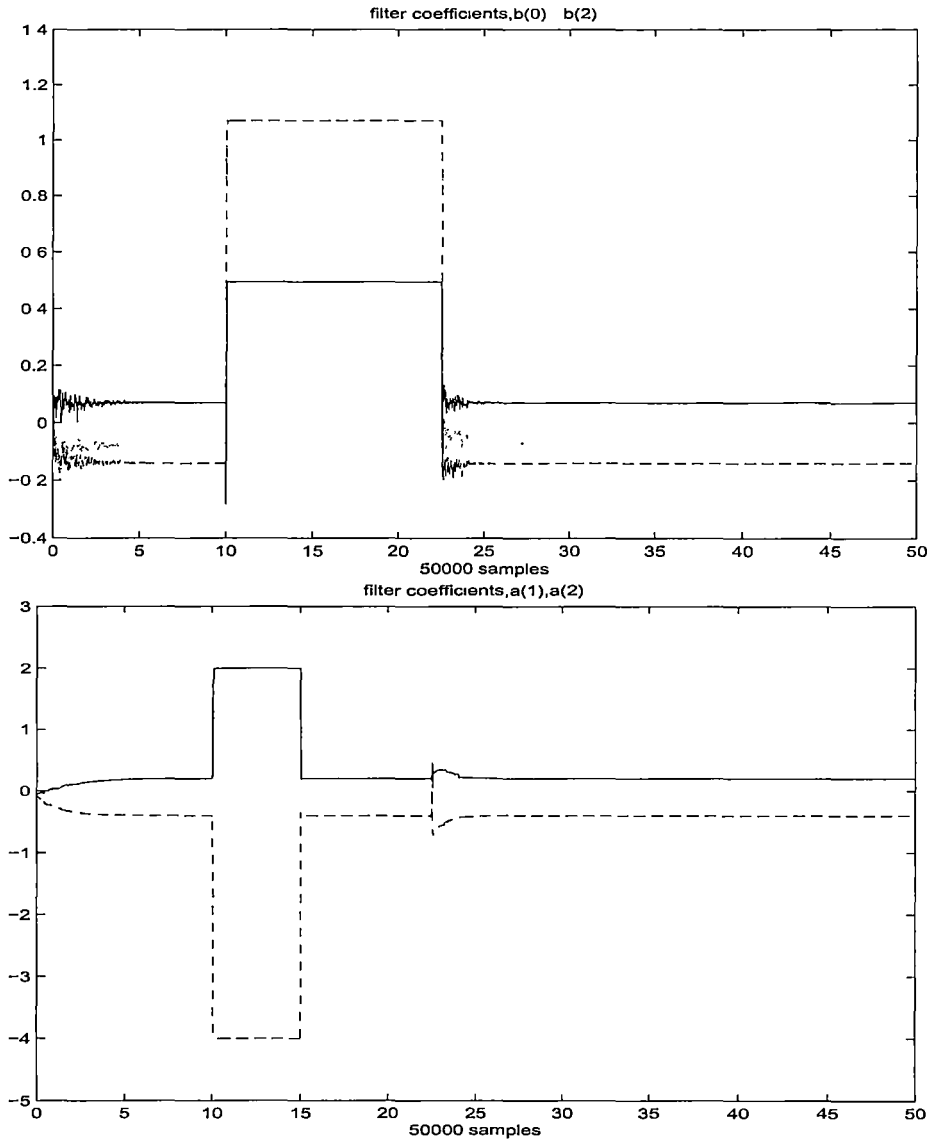


Figure 3.18: LAF-IIR filter- coefficients, $b_i(k)$, $a_i(k)$

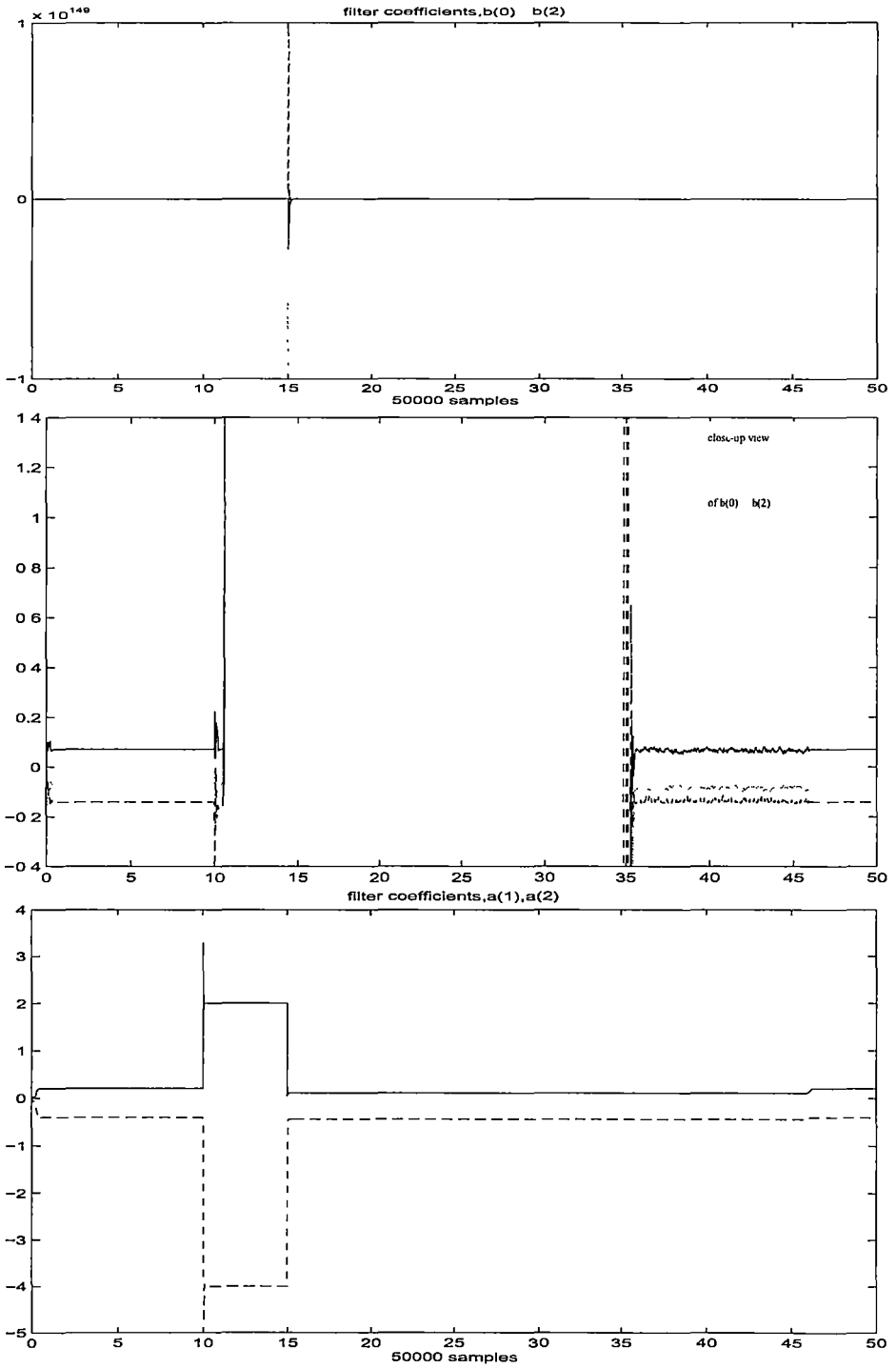
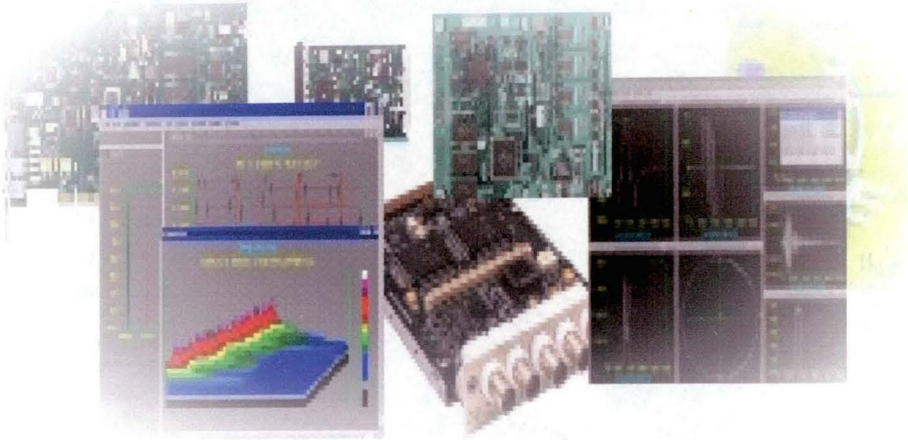


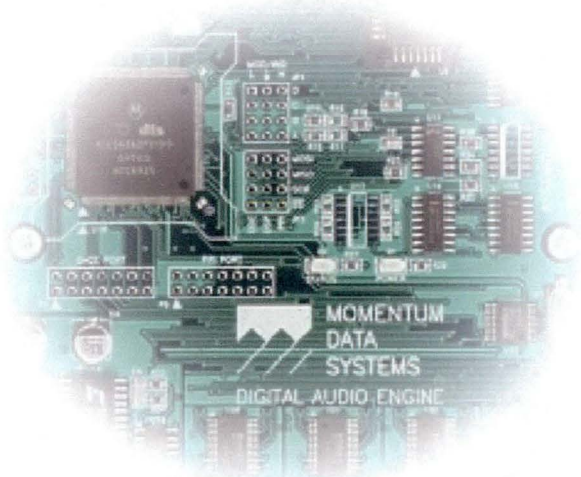
Figure 3.19: RLS (IIR) filter- coefficients, $b_i(k)$-*y axis* x10^{149}, close up view of $b_i(k)$ and coefficients of $a_i(k)$

3.8 Conclusion

This chapter has shown that Lyapunov stability theory has provided an efficient optimization method for the adaptive filter designs in the state space. We have discussed the convergence rate of the Lyapunov filter and the convergence region of the Lyapunov filter with the modified adaptive gain in order to avoid the singularity. Furthermore, the realizations of two adaptive filters, Lyapunov adaptive FIR and IIR filters have been developed. The design and analysis of LAF are extremely simplified and the stability concerns for the adaptive algorithm is guaranteed by the Lyapunov stability theory. Simulation examples have demonstrated the excellent convergence property and robustness to additive noise based on the new filter designs. The further research based on this chapter is to use different Lyapunov functions and different adaptive laws to further improve the convergence properties and the robustness properties of the Lyapunov filters with respect to the bounded random disturbances. In conclusion, the LAF has provided a new option to adaptive filtering and hopefully suggested a new research area of adaptive signal processing with Lyapunov stability theory.

Chapter 4





Radial Basis Function (RBF) Neural Network-based Adaptive Filtering

Chapter 4

Radial Basis Function (RBF) Neural Network-based Adaptive Filtering

4.1 Introduction

During the past decade neural networks have begun to find wide applicability in diverse aspects of signal processing, for example, filtering, parameter estimation, signal detection, system identification, pattern recognition, signal reconstruction, time series analysis, signal compression, and signal transmission [26]. The signals concerned include audio, video, speech, image, communication and others. The key features of neural networks involved in signal processing are their asynchronous parallel and distributed processing, nonlinear dynamics, global interconnection of network elements, self-organization and high-speed computational capability. With these features, neural networks can provide very powerful means for solving problems encountered in signal processing, especially in nonlinear adaptive filtering. Specific research works on neural networks for adaptive filtering can be found in [27]-[30].

While the majority of the research is directed towards a better architecture for neural networks, training algorithm or learning in the neural networks is another important research topic. In last few years, many researchers have focused their efforts on devising efficient algorithms, mainly based on gradient search methods. One of the potential problem, which is likely to affect practical applications, is that the learning process may be seriously plagued by the presence of stationary points in the cost function. In general, there is no reason to exclude the presence of stationary points

that may also be local minima. Obviously, this does not mean that no learning procedure can effectively find optimal solution, but, if the cost function has many local minima, devising an effective learning algorithm may be very difficult. On the other hand, for local minima free cost functions, simple gradient descent algorithms allow to discover optimal solutions with a relatively limited computational burden. These have motivated either the research on conditions for guaranteeing the absence of local minima [31]-[32] or the research on efficient and less computational burden algorithms to find the optimal solution [33]-[35]. Many authors [31]-[32] have analyzed the problem of optimal learning in neural networks by proposing some sufficient conditions which guarantee local minima free error surfaces. Some authors have proposed more computational complexity techniques such as genetic algorithms, learning automata and simulated annealing [33]-[35].

Recently, many researchers have used the *Radial Basis Function* (RBF) neural networks for a wide range of applications because of the distinctive properties of best approximation, simple network structure and training procedures. Authors in [32] have analyzed the problem of optimal learning in the RBF neural networks and proven that the attached cost function is local minima free under the assumption in [32]. However, the conditions that guarantee the local minima free problems in [32] are no longer applied when the feedback is considered in the RBF network. Therefore, after the cost function of the error is selected, the surface of the cost function in the parameter space is fixed. The search of the optimum parameters in the parameter space may stop at some local minimum because of the arbitrary initial condition of system states. Therefore the RBF neural networks with some gradient search-based algorithms may not give good performance.

To overcome the above problems, we propose two realizations of the Lyapunov adaptive filters using RBF neural networks. The FIR (finite impulse response) and IIR (infinite impulse response) filters are configured as feedforward and recurrent RBF networks respectively. It is shown in [20] that a Lyapunov function of the error between the desired signal and the RBF neural network output is defined, the weights of the RBF neural filter are then adaptively adjusted based on the *Lyapunov Theory-based adaptive filtering* (LAF) in Chapter 3, so that the error can asymptotically converge to zero. Unlike many adaptive neural filtering schemes using gradient

search in the parameter space, the selected Lyapunov function for the adaptive RBF filter has a unique global minimum in the state space. By properly choosing the weights update law in Lyapunov sense, the output of the adaptive RBF neural filter can asymptotically converge to the desired reference signal. Thus the local minima problem occurred in the gradient search-based adaptive filters is avoided,. Although the input signal of the RBF neural filter is disturbed by the bounded random noises, only the input and the output measurements are needed for the design of the RBF neural filters. Hence the proposed scheme is independent of the statistical properties of the input signals.

This chapter is organized as follows. In section 4.2, the realization of Lyapunov FIR filter using RBF neural network is proposed. The idea is extended to the nonlinear recurrent RBF IIR filter in section 4.3. The theoretical derivation is further supported by the simulation examples in the section 4.4. Finally, the concluding remark is presented in the last section of this chapter.

4.2 The Realization of Lyapunov FIR (Finite Impulse Response) Filters Using Feedforward RBF Neural Networks

Feedforward layered neural network or *multilayer perceptron* (MLP) has increasingly been used in many areas of signal processing. One of the disadvantages of MLP is that they are highly nonlinear in parameters. Learning must be based on nonlinear optimization techniques. The parameter estimate may be trapped at a local minimum of the chosen optimization criterion during the learning procedure when a gradient descent algorithm such as *backpropagation* (BP) is used. Other optimization techniques [33]-[35] are capable of achieving a global minimum but they require extensive computation. An alternative choice of highly nonlinear MLP is the RBF neural networks. The RBF network can be regarded as a special two layer network which is linear in the parameters by fixing all RBF centers and nonlinearities in the hidden layer. The output layer then implements a linear combiner on this new space and the only adjustable parameters are the weights of this linear combiner. These parameters can therefore be determined using linear algorithms [1]-[5], which is an important advantage of the RBF networks.

A feedforward RBF Lyapunov FIR filter is shown in *Figure 4.1.* The output of the RBF FIR filter can be expressed as

$$y(k) = \sum_{i=1}^{N} w_i(k)\phi_i(k) \tag{4.1}$$

or

$$y(k) = W^T(k)\Phi(k) \tag{4.2}$$

where $W(k) = [w_1(k), w_2(k), \cdots, w_N(k)]^T$ (4.3)

$\Phi(k) = [\phi_1(k), \phi_2(k), ..., \phi_N(k)]^T$ (4.4)

$\phi(k)$ is the Gaussian type of functions defined as

$$\phi_i(k) = \exp\left(-\frac{\|X(k) - c_i\|}{\sigma_i^2}\right) \qquad i = 1, 2, 3, ...N \tag{4.5}$$

and $X(k) = [x(k), x(k-1), ..., x(k-N)]^T$, (4.6)

c_i is the center vector and σ_i is the width of Gaussian function. The width is controlled by the noise variance σ_n^2 and is usually set at $\sigma_i = 2\sigma_n^2$.

Using the results of theorems 3.1-3.3 in Chapter 3, we have the following updated law for Lyapunov RBF FIR adaptive filter:

$$W(k) = W(k - 1) + g(k)\alpha(k) \tag{4.7}$$

$$\alpha(k) = d(k) - W^T(k-1)\Phi(k) \tag{4.8}$$

$$g(k) = \frac{\Phi(k)}{\|\Phi(k)\|^2}\left(1 - \kappa \frac{|e(k-1)|}{|\alpha(k)|}\right) \tag{4.9}$$

or $$g(k) = \frac{\Phi(k)}{\|\Phi(k)\|^2 + \lambda_1}\left(1 - \kappa \frac{|e(k-1)|}{\lambda_2 + |\alpha(k)|}\right) \tag{4.10}$$

4.2.1 Design of the Adaptive Filter Using RBF Neural Network and Lyapunov Theory

The design of the adaptive RBF neural filter is similar to the *Lyapunov Theory-based adaptive filtering* (LAF) and can be described by Theorem 4.1:

Theorem 4.1: For the given desired response $d(k)$, if the weight vector $W(k)$ of the filter $y(k) = W^T(k)\Phi(k)$ is updated as follows

$$W(k) = W(k-1) + g(k)\alpha(k) \tag{4.11}$$

$$g(k) = \frac{\Phi(k)}{\| \Phi(k) \|^2}\left(1 - \kappa\frac{|e(k-1)|}{|\alpha(k)|}\right) \tag{4.12}$$

where $\alpha(k) = d(k) - W^T(k-1)\Phi(k)$ in the expression (4.8), $0 \le \kappa < 1$, then the error $e(k) = d(k) - y(k)$ asymptotically converges to zero.

Proof: Define a Lyapunov function

$$V(k) = e^2(k) \tag{4.13}$$

Then,

$$\begin{aligned}
\Delta V(k) &= V(k) - V(k-1) \\
&= e^2(k) - e^2(k-1) \\
&= (d(k) - W^T(k)\Phi(k))^2 - e^2(k-1) \\
&= (d(k) - (W^T(k-1) + g^T(k)\alpha(k))\Phi(k))^2 - e^2(k-1) \\
&= (d(k) - W^T(k-1)\Phi(k) - g^T(k)\alpha(k)\Phi(k))^2 - e^2(k-1) \\
&= (\alpha(k) - g(k)\alpha(k)\Phi(k))^2 - e^2(k-1) \\
&= \alpha^2(k)\left(1 - g^T(k)\Phi(k)\right)^2 - e^2(k-1)
\end{aligned} \tag{4.14}$$

Using the expression (4.12) in the expression (4.14), we have

$$\Delta V(k) = -(1-\kappa^2)e^2(k-1) < 0 \tag{4.15}$$

With reference to Lyapunov second method [20] or Chapter 3, the error $e(k)$ will converge to zero asymptotically.

Remark 4.1: It is easy to see that the stability analysis of the error dynamics, convergence analysis of the Lyapunov RBF FIR adaptive filter are same as the ones given in *Theorems 3.1, 3.1, 3.3* in Chapter 3 if we replace $X(k)$ by $\Phi(k)$.

***Remark 4.2*:** The proposed adaptive algorithm has possessed the similar properties of the LAF in Chapter 3. The designed feedforward RBF FIR filter is independent of the stochastic properties of signals. Based on the observations and a collection of desired response, the weights of the feedforward RBF neural network are updated in Lyapunov sense so that the error between the desired response and the RBF neural filter output can asymptotically converge to zero. The stability of the error dynamics is guaranteed based on the Lyapunov stability theory [20]. The error convergence rate relies on the constant κ in the expression (4.12). The smaller value of constant κ gives faster the error convergence rate. Smaller λ_1 and λ_2 values contribute smaller error.

Remark 4.3: For the center vectors, the simplest technique involves choosing these vectors randomly from a subset of the available sample vectors. However, in such a case the number of hidden neurons needs to be relatively large to cover the entire input domain, *k-means clustering* [30],[36],[37] algorithm based on the non-hierarchical clustering methods can be employed to update the centers.

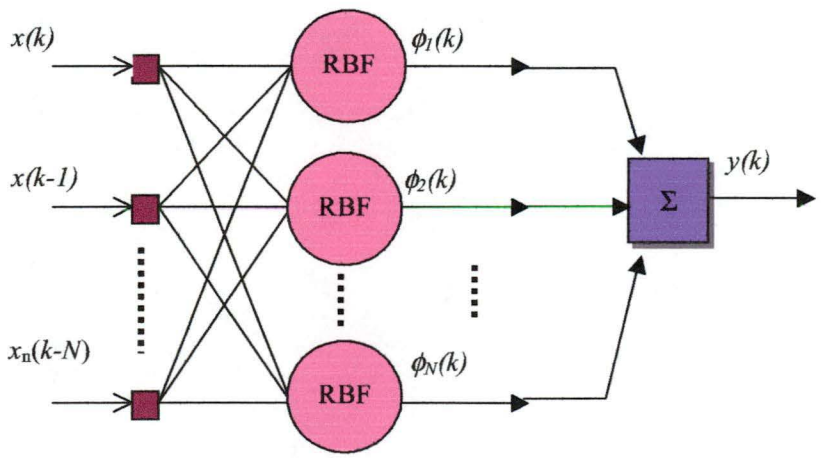


Figure 4.1: Nonlinear Feedforward RBF FIR Filter

4.3 The Realization of Lyapunov IIR (Infinite Impulse Response) Filters Using Recurrent RBF Neural Networks

Recently, recurrent neural networks have been attracting much attention because of their attracting capability to exhibit dynamic behavior. They represent a very powerful computational model, but designing proper architecture for a given problem and devising effective learning procedures are very challenging tasks. Various recurrent neural network architectures and learning algorithms have been developed [38],[39],[30],[31]. Extension of the *backpropagation* (BP) to the recurrent networks was first proposed in [40]. In general, two popular approaches exist for recurrent networks: *Backpropagation through time* (BPTT) [28],[40],[41] and *real-time recurrent learning* (RTRL) [42],[39],[28]. Different modified BPTT algorithms have been derived [42],[41],[28]. C. Paolo [42] developed two new gradient-based procedures called *recursive backpropagation* (RBP) and a on-line version, casual RBP (CRBP) for *locally recurrent neural networks*. Pearlmutter [38] and Williams [41] presented alternative methods, designed to achieve results similar to those of BPTT, using a different computational strategy. However, it was reported in [43] that the Williams-Zipser architecture typically suffers from a lack of stability, slow convergence and the system may converge to a local minimum in the parameter space. A.C Tsoi and Back A.D [43]-[44] have introduced the locally recurrent globally feedforward (LRGF) networks architecture with local synapse feedback that they called an IIR synapse MLP. A first order learning rule minimizing a mean square error criterion was derived and the weight changes could be adjusted using simple gradient method.

Up to this point, we may notice that the algorithms used in the aforementioned recurrent networks are the gradient descent methods or other gradient based optimization technique such as conjugate-gradient. Therefore, they have potential to settle in the sub-optimal solution.

In this section, we present the realization of Lyapunov IIR Filters Using RBF neural networks. The nonlinear *output error* IIR filter is realized using a recurrent RBF

network. The basic structure of a recurrent RBF Lyapunov IIR filter is given in *Figure 4.2*. The output of the RBF IIR filter is written as

$$y(k)=\sum_{i=1}^{N} w_i(k)\phi_i(k)+\sum_{j=1}^{M} w_{N+j}(k)\phi_{N+j}(k) \tag{4.16}$$

or

$$y(k) = \theta^T(k)\Phi(k) \tag{4.17}$$

where $\theta(k)=[w_1(k),\ w_2(k),\ \cdots,\ w_N(k), w_{N+1}(k),\ w_{N+2}(k),\ \cdots,\ w_{N+J}(k),]^T$ (4.18)

$$\Phi(k) = [\phi_1(k),\ \phi_2(k),\ ...,\ \phi_N(k), \phi_{N+1}(k),\ \phi_{N+2}(k),\ ...,\ \phi_{N+M}(k)]^T \tag{4.19}$$

$$X(k) = [x(k),\ x(k-1),\ ...,\ x(k-N), y(k-1),\ y(k-2),\ ..., y(k-N)]^T. \tag{4.20}$$

and $\phi_i(k)$ is defined in the expression (4.5), but $X(k)$ in (4.20) is used instead of $X(k)$ in (4.6).

Using the results in Chapter 3 or Section 4.2 in this Chapter, the network weights can be updated as follows:

$$W(k) = W(k-1) + g(k)\alpha(k) \tag{4.21}$$

$$\alpha(k) = d(k) - W^T(k-1)\Phi(k) \tag{4.22}$$

$$g(k)=\frac{\Phi(k)}{\|\Phi(k)\|^2}\left(1-\kappa\ \frac{|e(k-1)|}{|\alpha(k)|}\right) \tag{4.23}$$

and the modified $g(k)$ is given by

$$g(k)=\frac{\Phi(k)}{\|\Phi(k)\|^2+\lambda_1}\left(1-\kappa\frac{|e(k-1)|}{\lambda_2+|\alpha(k)|}\right) \tag{4.24}$$

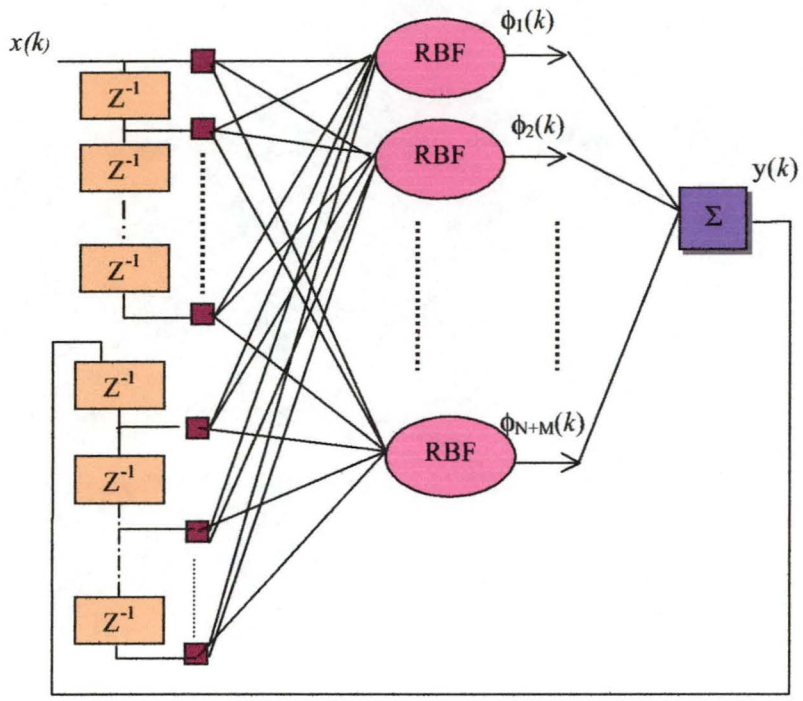


Figure 4.2: Nonlinear Recurrent RBF IIR Filter

***Remark 4.4*:** The advantage of the RBF IIR design compared to the RBF FIR design is that the use of lagged output variables reduces the number of coefficients that are required for an effective design. However, this is offset by the similar case in the output error IIR filter that the stability of the filter is no longer guaranteed. Design of the adaptive algorithm for RBF IIR filter can provide a solution to this problem. The stability of the error dynamic is guaranteed by the Lyapunov stability theory.

***Remark 4.5*:** The local minima free condition in [32] for the feedforward RBF network is also no longer assured if the feedback is considered. Therefore the adaptive algorithms [1]-[3] used in the feedforward RBF network may not provide a good performance in the recurrent RBF network. As explained in Chapter 3, a Lyapunov function of the error between the desired signal and the RBF neural network output is defined. The selected Lyapunov function for the RBF filter design has a unique global minimum. By properly selecting the weights update law in

Lyapunov sense, the output of the RBF filter can asymptotically converge to the desired signal. Using this method, the local minima problem occurred in the gradient search-based methods can be prevented. In addition, the design is independent of the stochastic properties of the input disturbances.

4.4 Simulation Examples

In this section, the following two simulation examples are presented to illustrate the performance of the proposed RBF Lyapunov adaptive filters.

Example 1: Feedforward RBF FIR filter

In this example, we will compare the proposed scheme with other existing adaptive filtering schemes to show the robustness and effectiveness.

1. The feedforward RBF FIR filter with the LAF (1) algorithm ($\kappa = \lambda_1 = \lambda_2 = 0.01$)
2. The feedforward RBF FIR filter with the LAF (2) algorithm ($\kappa = \lambda_1 = \lambda_2 = 0.001$)
3. The feedforward RBF FIR filter with the LAF (3) algorithm ($\kappa = \lambda_1 = \lambda_2 = 0.0001$)
4. The feedforward RBF with the RLS algorithm (forgetting factor $\rho = 0.9$)
5. The feedforward RBF with the RLS algorithm (forgetting factor $\rho = 0.1$)
6. The feedforward RBF with the LMS algorithm
7. The feedforward MLP with the BP algorithm

The RBF neural networks used in this simulation have 3 input nodes, 3 hidden nodes and 1 output node. Centers for the RBF network are selected randomly from the input subset because of the small input domain. The MLP has the same structure, but the input-hidden and hidden-output layers have connection weights.

In the first case, no additive noise is considered in the simulation. *Figure 4.3* has revealed the comparison of the feedforward RBF FIR filter output $y(k)$ and the desired signal, $d(k)$. The square error, $e^2(k)$ is illustrated in *Figure 4.4* and the weights of RBF network are plotted in *Figure 4.5*. *Figure 4.6* shows the average square error of each iteration for different neural filters and their MSEs (mean square

error) are summarized in *Table 4.1*. Although the selected centers may not be optimal for each case, these results have illustrated the proposed RBF filter is outperform compared with others.

In the second case, the signal is corrupted by a uniformly distributed white noise sequence, *n(k)* varying in the range [0,0.5] and gives SNR (signal to noise ratio) ≈11 dB approximately. The simulation results of their MSEs are tabulated in *Table 4.2*. From the simulation results, the effect of additive noise is reduced greatly in the proposed RBF filter. The sigmoid function in the MLP can suppress the noise effects by using their saturation regions, thus the MSE without noise is similar to the MSE with additive noise. The simulation example has verified that the smaller λ_1 and λ_2, smaller the error *e(k)* and the smaller constant κ gives faster error convergence. The RBF network trained by the RLS with a smaller forgetting factor ($\rho = 0.1$) can give better performance compared with the RLS with a larger forgetting factor ($\rho = 0.9$). However the weights have large variation in magnitude and this is not desired in many applications.

In summary, the simulation results have revealed that the proposed RBF filter has better performance in terms of error convergence, tracking ability and resistance of additive noise.

Example 2: Recurrent RBF IIR filter

In this example, we compare the following adaptive IIR filters

The recurrent RBF IIR filter with the LAF (1) algorithm ($\kappa = \lambda_1 = \lambda_2 = 0.01$)
The recurrent RBF IIR filter with the LAF (2) algorithm ($\kappa = \lambda_1 = \lambda_2 = 0.001$)
The recurrent RBF IIR filter with the LAF (3) algorithm ($\kappa = \lambda_1 = \lambda_2 = 0.0001$)
The recurrent MLP with the BPTT (Backpropagation through time) algorithm
The recurrent MLP with the RTRL (Real time recurrent learning) algorithm

The RBF neural networks used in this simulation have 5 input nodes, 5 hidden nodes and a output node. A feedback is connected from the output layer to two nodes in the input layer. The MLP has the same number of nodes and feedback connections as the

RBF network. The average square error for each iteration of different neural filters are illustrated in *Figure 4.7* (without noise) and their MSEs are summarized *in Table 4.3* and *Table 4.4*. Again these simulations are intended to show the proposed scheme can have good performance. Because of the difference in the RBF and MLP architectures, it is quite difficult to find a basis for comparison. The one gives smaller MSE or average square error would be considered as a better method. Simulations of the recurrent RBF with RLS or LMS algorithms have been performed but the error divergence has been observed for both cases. In summary, the proposed RBF filter exhibits excellent performance with the relatively simple recurrent network architecture

Table 4.1: Feedforward Neural Network Filters (No Additive Noise)

	RBF-LAF ($\kappa=\lambda_1=\lambda_2$ =0.01)	RBF-LAF ($\kappa=\lambda_1=\lambda_2$ =0.001)	RBF-LAF ($\kappa=\lambda_1=\lambda_2$ =0.0001)	RBF-RLS (ρ = 0 9)	RBF-RLS (ρ = 0.1)	RBF-LMS	MLP-BP
MSE	5.16×10^{-6}	5.51×10^{-8}	5.54×10^{-10}	8.4×10^{-3}	1.32×10^{-5}	0.0222	0.0211

Table 4.2: Feedforward Neural Network Filters (With Additive Noise)

	RBF-LAF ($\kappa=\lambda_1=\lambda_2$ =0.01)	RBF-LAF ($\kappa=\lambda_1=\lambda_2$ =0.001)	RBF-LAF ($\kappa=\lambda_1=\lambda_2$ =0.0001)	RBF-RLS (ρ = 0.9)	RBF-RLS (ρ = 0.1)	RBF-LMS	MLP-BP
MSE	8.92×10^{-5}	2.9×10^{-7}	1.61×10^{-8}	0.0168	1.36×10^{-3}	0.05	0.0211

Table 4.3: Recurrent Neural Network Filters (No Additive Noise)

	RBF-LAF ($\kappa=\lambda_1=\lambda_2$=0.01)	RBF-LAF ($\kappa=\lambda_1=\lambda_2$=0.001)	RBF-LAF ($\kappa=\lambda_1=\lambda_2$=0.0001)	MLP-BPTT	MLP-RTRL
MSE	4.63×10^{-6}	4.9×10^{-8}	5.0×10^{-10}	0.0206	0.0171

Table 4.4: Recurrent Neural Network Filters (With Additive Noise)

	RBF-LAF ($\kappa=\lambda_1=\lambda_2$=0.01)	RBF-LAF ($\kappa=\lambda_1=\lambda_2$=0.001)	RBF-LAF ($\kappa=\lambda_1=\lambda_2$=0 0001)	MLP-BPTT	MLP-RTRL
MSE	1.35×10^{-5}	1.2×10^{-7}	1.2×10^{-9}	0.0206	0.0171

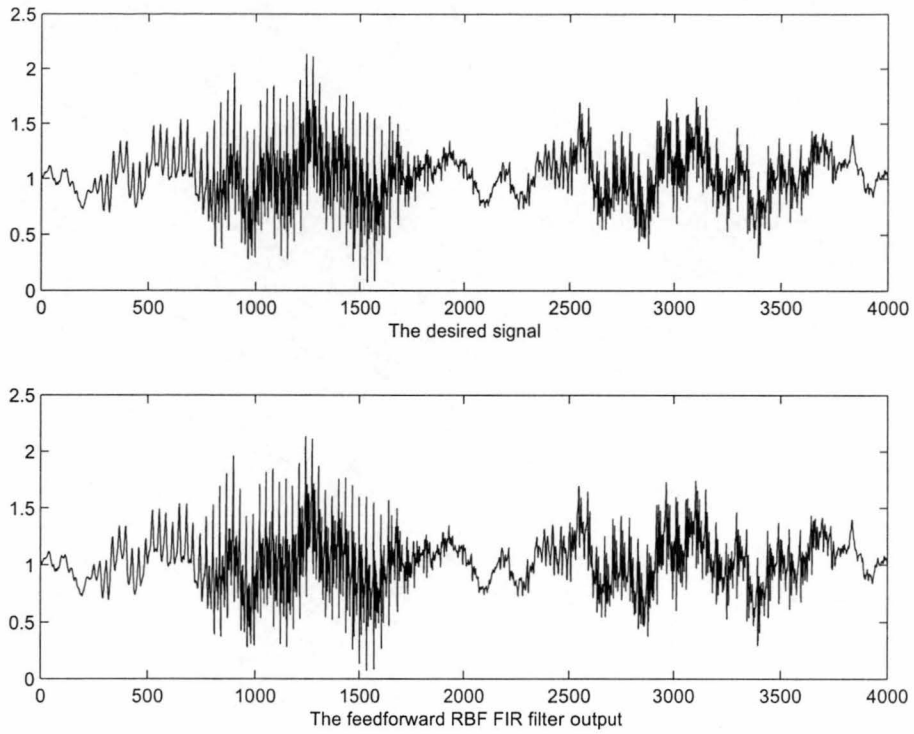


Figure 4.3: The feedforward RBF FIR filter output *y(k)* and the desired signal *d(k)*

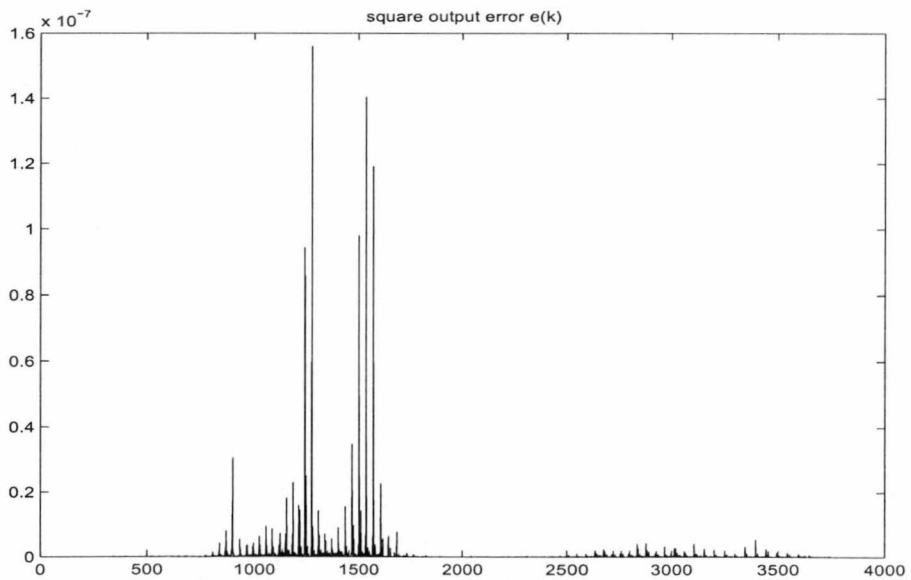


Figure 4.4 The square error, $e^2(k)$ of the feedforward RBF FIR filter

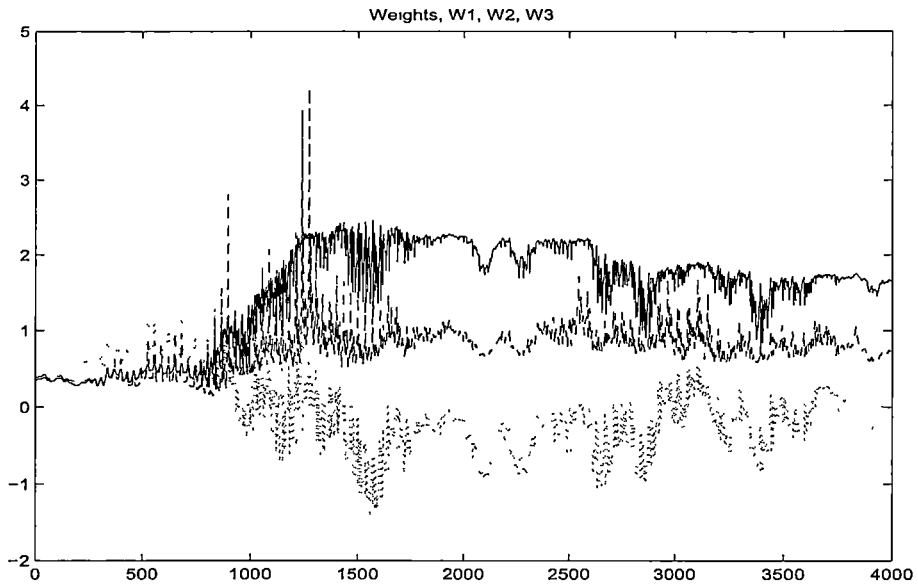


Figure 4.5: The weights of the feedforward RBF FIR filter

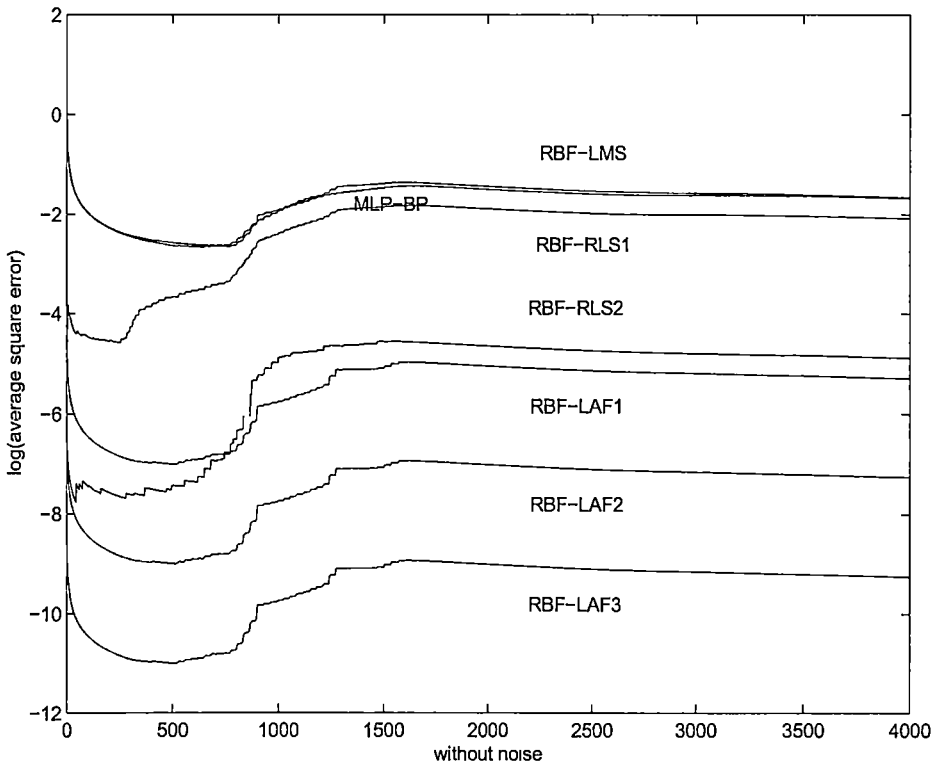


Figure 4.6: The average square error for feedforward neural network filters

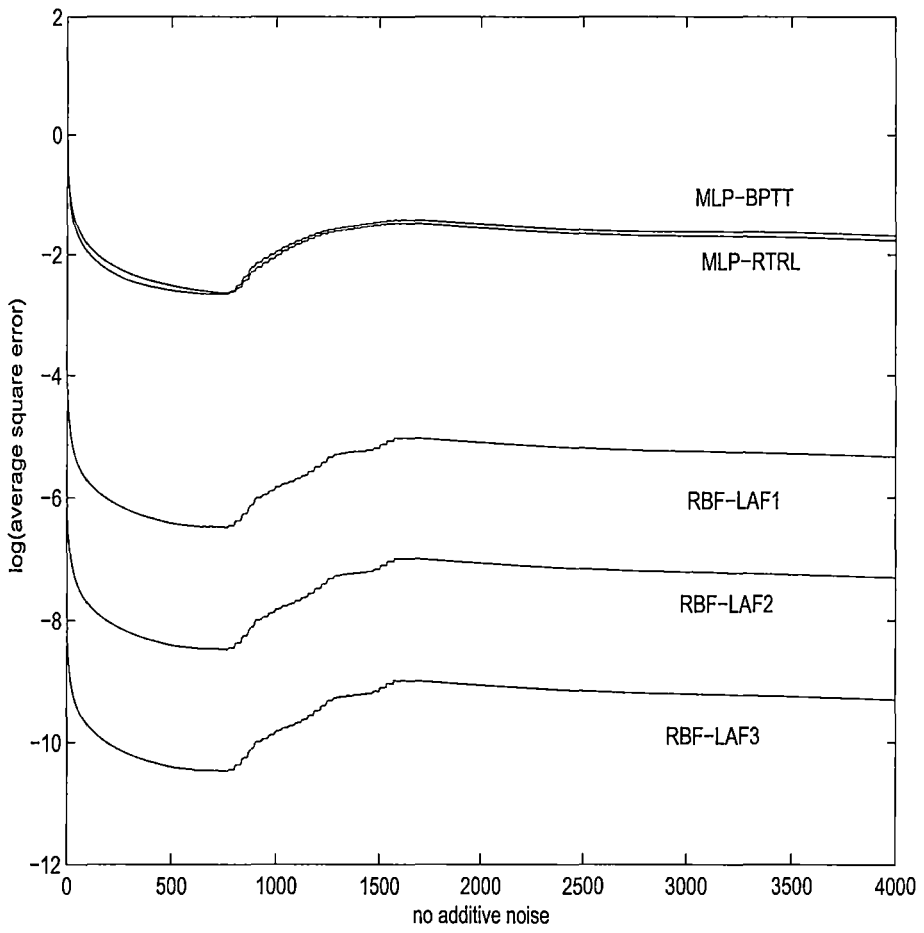
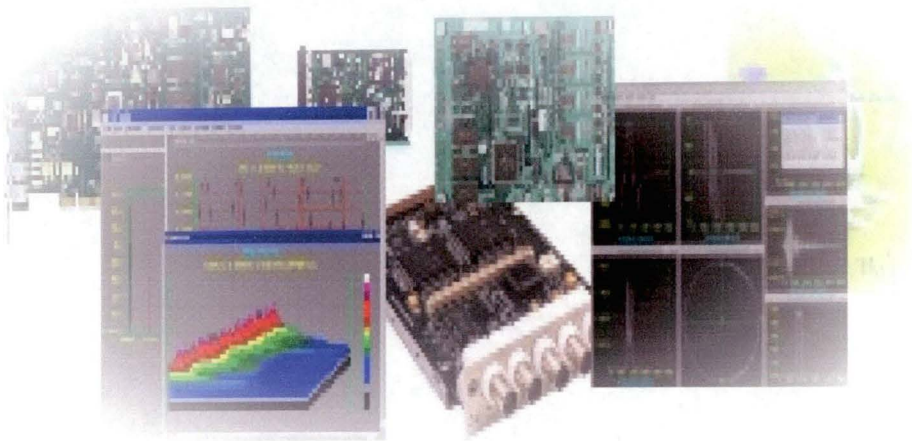


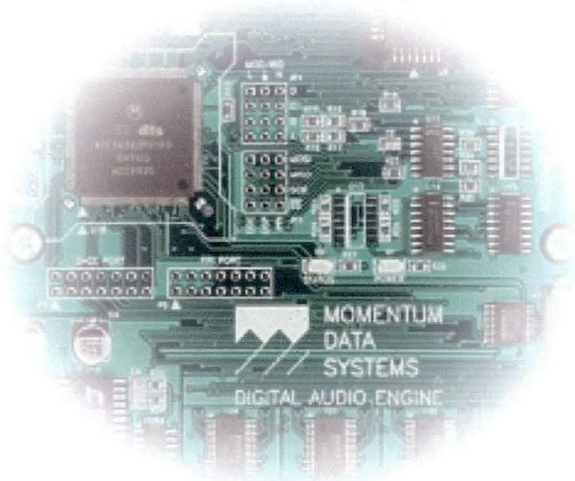
Figure 4.7: The average square error for recurrent neural network filters

4.5 Conclusion

This chapter has indicated that Lyapunov stability theory can provide an efficient training method for the adaptive RBF filter designs. New nonlinear FIR and IIR filter realizations based on the feedforward and recurrent RBF networks have been introduced for nonlinear adaptive filtering problem. The LAF idea is incorporated into the adaptive algorithms for the nonlinear adaptive RBF FIR and IIR filters. Thus these adaptive algorithms possess the properties of the LAF algorithm and the RBF network. The local minima problem occurred in the gradient search based adaptive algorithm can be avoided by using the proposed scheme. The network weights updated strategy is independent of signal statistical properties because only the desired response and input observations are needed. Theoretical analysis and simulations have indicated the proposed method can offer good performance in terms of stability, tracking and convergence properties. In conclusion, the RBF networks based on LAF has provided a new and alternative approach to conventional adaptive filtering problem. Hopefully this will suggest a new future research of adaptive signal processing using Lyapunov theory. The further research is to use different Lyapunov functions and different weight laws to further improve the convergence properties and the robustness properties of the RBF filters with respect to the bounded random disturbances.

Chapter 5





Fuzzy Adaptive Filters Using Lyapunov Theory-based Adaptive Filtering

Chapter 5

Fuzzy Adaptive Filters Using Lyapunov Theory-based Adaptive Filtering

5.1 Introduction

Filters are information processor. In practice, information usually comes from two sources: sensors which provide numerical data associated with a problem, and human experts who provide linguistic descriptions (often in the form of fuzzy IF-THEN rules) about the problem. Existing filters can only processing numerical data, whereas existing expert systems can only make use of linguistic information. Therefore, their successful applications are limited to problems where either linguistic rules or numerical data do not play a critical role. There are, however, a large number of practical problems in economics, seismology, management, and so on, where both linguistic and numerical information are critical.

At present, when we face such problems, we use linguistic information, consciously or unconsciously, in the choice among different filters, the evaluation of filter performance, the choice of filter orders, the interpretation of filtering results, and so on. There are serious limitations to use linguistic information in this way because for most practical problems the linguistic information (in its natural form) is not about what kind of filter should be chosen or what the order of the filter should be, but is in the form of fuzzy IF-THEN rules.

In this situation, fuzzy logic has stirred a great deal of excitement, since it allows for the simple inclusion of heuristic knowledge about how to filter the noise rather than

requiring exact mathematical model. Furthermore, the fuzzy adaptive filter has the universal approximation ability in nonlinear problems [45]-[48]. The fuzzy rules come either from human experts or by matching input-output pairs through an adaptation procedure. Authors of [45] have presented a fuzzy adaptive filter that is constructed from a set of changeable fuzzy IF-THEN rules to minimize some criterion functions. These fuzzy adaptive filters parameters are updated by *recursive least square* (RLS) and *least mean square* (LMS) algorithms. They have mentioned the computation complexity involved in the RLS fuzzy filter is highly parallelizable and the RLS fuzzy filter might not be able to be used in some practical situations where the computing power is limited. In contrast, the LMS fuzzy filter has suffered the problem encountered in the LMS filter such as slow error convergence.

The purpose of this chapter is to develop new kinds of nonlinear adaptive filters, which we refer to as *fuzzy adaptive filters.* First, a *fuzzy gain Lyapunov adaptive filter* for nonlinear adaptive filtering is proposed. This scheme is designed based on the LAF [19] in Chapter 3 and fuzzy logic is introduced to the filter design. It incorporates fuzzy logic to the LAF by the use of a set of Lyapunov sense fuzzy if-then rules. Given the input signal and its squared norm, these rules are then used to determine the adaptive gain to update the filter parameters so that the error converges to zero asymptotically. An additional computational cost is incurred in the fuzzification, inference and defuzzification modules, but these operations can be done very efficiently in the latest range of DSP. Simulation examples of the fuzzy gain Lyapunov adaptive filter are performed to support the theoretical results. Comparisons with the numerical adaptive filters using the LAF and RLS algorithms are also presented.

The second fuzzy adaptive filter is named *LAF fuzzy adaptive filter.* This fuzzy adaptive filter is constructed from a set of changeable fuzzy IF-THEN rules. The adaptive algorithm, *Lyapunov theory-based adaptive filtering* (LAF) is used to update the parameter of the membership functions so that the dynamic error between the filter output and the desired response converges to zero asymptotically. Therefore, the most significant advantage of the fuzzy filter compared to the conventional filters is that linguistic information from human experts (in the form of fuzzy IF-THEN rules) can be incorporated into the filter. If no linguistic information

is available, the fuzzy adaptive filters become well-defined nonlinear adaptive filters. The fuzzy adaptive filter has preserved the properties of LAF in Chapter 3 such as fast convergence, highly stable and independent of the signal's stochastic properties. The computational complexity involved is less than that of the RLS fuzzy filter in [45]. Simulation examples of LAF fuzzy adaptive filter are performed to support the theoretical results.

This chapter is organized as follows. In section 5.2, the fuzzy gain Lyapunov adaptive filter for nonlinear adaptive filtering is proposed. Section 5.3 presents design methodology of FIS (Fuzzy Inference System) of fuzzy gain Lyapunov adaptive filter. The theoretical derivation of the fuzzy gain Lyapunov adaptive filter is further supported by the simulation examples in the section 5.4. Section 5.5 establishes the second fuzzy adaptive filter, LAF fuzzy adaptive filter. Design procedure of the LAF fuzzy adaptive filter and the simulation examples are presented in section 5.6 and section 5.7 respectively. Finally, the concluding remark is presented in the last section of this chapter.

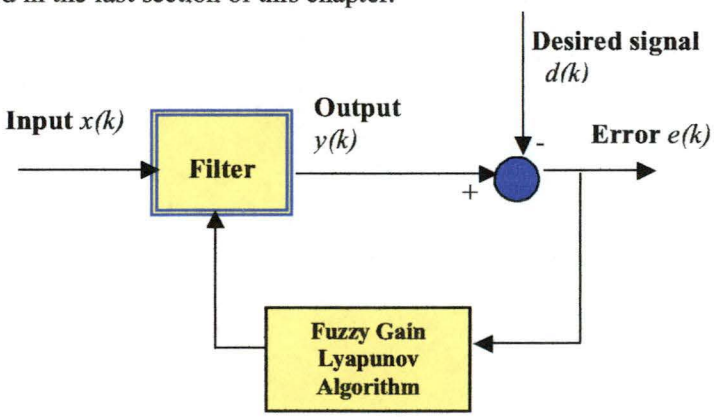


Figure 5.1: Fuzzy Gain Lyapunov Adaptive Filter Architecture

5.2 Fuzzy Gain Lyapunov Adaptive Filter

The advantages of the LAF scheme have been explained in Chapter 3. However, there are certain circumstances that adaptive filtering has to deal with many ambiguous situations. Therefore fuzzy logic is a useful mathematical tool for handling the ambiguity or uncertainty. In order to apply fuzzy theory to the adaptive filter, selecting the fuzzy rules and regions of membership function are fundamental and important tasks. The structure of the fuzzy gain Lyapunov adaptive filter is illustrated in *Figure 5.1*. *Figure 5.2* shows the fuzzy inference system (FIS) of the proposed fuzzy filter.

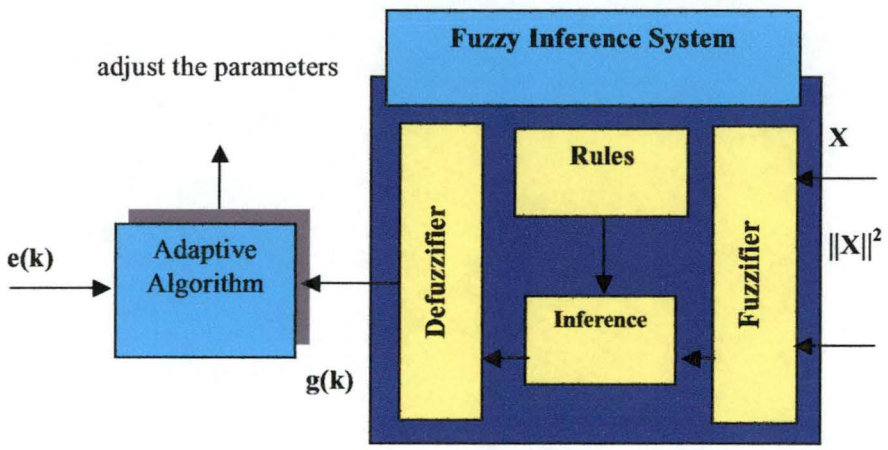


Figure 5.2: Adaptive Fuzzy Gain Algorithm

The expressions used to update the filter parameters are similar to the LAF in Chapter 3 and can be summarized as follows:

$$H(k) = H(k - 1) + g(k)\alpha(k) \tag{5.1}$$

$$\alpha(k) = d(k) - H^T(k-1)x(k) \tag{5.2}$$

$$g(k) = \frac{X(k)}{\| X(k) \|^2}\left(1 - \kappa\frac{|e(k-1)|}{|\alpha(k)|}\right) \tag{5.3}$$

or

$$g(k) = \frac{X(k)}{\lambda_1 + \| X(k) \|^2}\left(1 - \kappa\frac{|e(k-1)|}{\lambda_2 + |\alpha(k)|}\right) \tag{5.4}$$

where $0 \leq \kappa < 1$, and λ_1, λ_2 are small positive numbers.

However, the computation of the adaptation gain *g(k)* in (5.3) is totally a new approach in fuzzy gain Lyapunov adaptive filter. The adaptation gain *g(k)* in (5.3) is the crisp output of the FIS. This gain is adaptively adjusted so that error *e(k)* can converge and good performance can be achieved. In the following sections, the design of the fuzzy gain Lyapunov adaptive filter that the IF-THEN rules and MBFs in the FIS using on the LAF will be presented.

In the design of the fuzzy gain filter based on the LAF, *IF-THEN* fuzzy rules can be derived from (5.3). Rule matrix (*Table 5.1*) of adaptation gain is constructed based on $g = X / \| X \|^2$. For example, *IF X* is Z (input signal is zero or very small) *AND* $//X//^2$ is *ZI* (its squared norm value is zero or very small) *THEN g* is *ZERO* (the gain is approximate zero).

The final process of the *FIS* is to convert or defuzzify the aggregated fuzzy value for the adaptation gain into a crisp value to update the weight vector in (5.1). The design detail of FIS will be discussed in the next section. By designing the *IF-THEN rules* based on the rules matrix of the adaptation gain, the error *e(k)* can converge and good filtering performance is obtained.

5.3 Design Methodology of FIS of Fuzzy Gain Lyapunov Adaptive Filter

In order to apply fuzzy theory to the filter, selecting the fuzzy rules, regions of membership function are very important to achieve good performance. Some of the parameters and techniques used to implement the *FIS* are as follows: the selection of the types of membership function (*MBF*), the *MBF* parameters, fuzzy operators used, implication methods, aggregation methods and defuzzification schemas. The purpose of this section is to introduce useful directions in designing the fuzzy gain filter.

X(k) $\|X\|^2$	**NB** **Negative** **Big**	**NM** **Negative** **Medium**	**NS** **Negative** **Small**	**Z** **Zero**	**PS** **Positive** **Small**	**PM** **Positive** **Medium**	**PB** **Positive** **Big**
Z1				Zero			
PS1			S_2	Zero	S_5		
PM1		M_1	S_3	Zero	S_6	M_2	
PB1	S_0	S_1	S_4	Zero	S_7	S_8	S_9

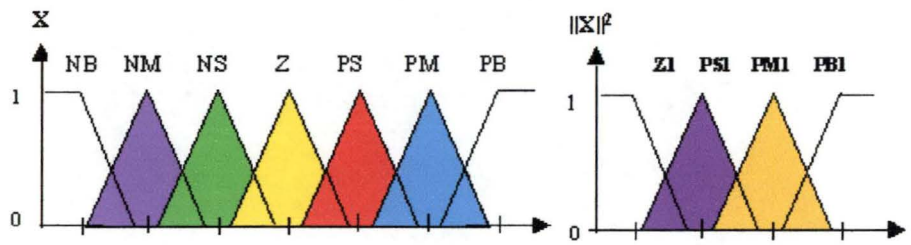
Table 5.1. Rules Matrix of the adaptation gain, *g(k)*

5.3.1 Determination of Fuzzy Sets For (1) The Input (X), (2) The Input Squared Norm ($||X||^2$), and (3) The Output (The Adaptive Gain)

Firstly, the input variables to the *FIS* (the input and its squared norm value) are converted to appropriate fuzzy sets via membership function (*MBF*). These fuzzy sets are used for partitioning the continuous domain of input and output variables into a small number of overlapping regions. These regions are labeled with linguistic terms such as '*Negative Big*', '*Negative Medium*', '*Negative Small*', '*Zero*', '*Positive Big*', '*Positive Medium*', '*Positive Small*'... etc as indicated in *Figure 5.3* for X and $//X//^2$ respectively. The task here is to locate the positioning of universe of discourse of these fuzzy sets.

The input limit for X can be obtained from observing the input numerical data. Seven *MBF*s (triangular/trapezoidal/etc) are selected to cover the entire universe of discourse as shown in *Figure 5.3*. Selection of the type of MBFs depends on the specific application or input signal. Then centroids for (*NB...PB*) are selected. The bases of *MBF*s cover the neighboring centroid as shown in *Figure 5.3*. The *NB, NM, NS* are just the mirror image of the positive *MBF*s shown in *Figure 5.3*.

The regions of the MBFs of the adaptation gain can be determined from observing the adaptation gain numerical data. Thirteen *MBFs* (triangular/trapezoidal/etc) are chosen to cover the entire universe of discourse as illustrated in *Figure 5.4*. Selection of the type of MBFs depends on the specific application or input signal again. Then centroids for (S_0 ,..., S_9 , M_0 , M_1 , *Zero*) are selected. The bases of *MBFs* cover the neighboring centroid as shown in *Figure 5.4*.



a) input value X *b) $||X||^2$*

Figure 5.3: MBFs spread over their respective universes of discourse

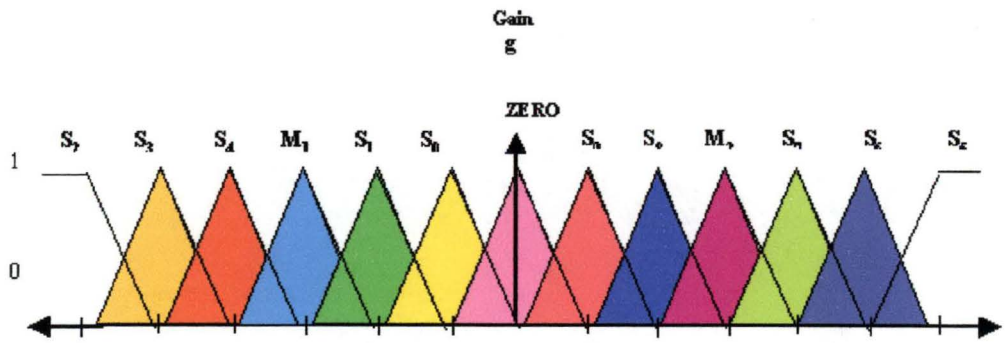


Figure 5.4: MBFs of the adaptive gain spread over its bound

5.3.2 Fuzzification of Inputs

The *FIS* takes in two fuzzy inputs: X and $||X||$ at time k. Then it determines the respective degree to which they belong to each of the appropriate fuzzy sets via

triangular/trapezoidal *MBF*s. The crisp numerical inputs must be limited to their respective universe of discourse of the input variables. The output of the fuzzification process is a fuzzy degree of membership between 0 and 1.

5.3.3 Fuzzy Rule Selection

The second step is to construct a set of fuzzy *IF-THEN* rules of the following form. This step has been metioned in the section 5.2. For example, *IF* input (X) is Z and $\|X\|^2$ is *Z1* then the adaptation gain is *ZERO*. *Tables 5.1* shows the fuzzy rules for the adaptive fuzzy filter. These *IF-THEN* fuzzy rules have simply been derived from the adaptive gain in (5.3). The rules matrix is constructed in *Table 5.1*. Different weights can be assigned to the different rules to emphasize the importance of a particular rule in a specific application.

5.3.4 Fuzzy Operators

In the fuzzy gain Lyapunov adaptive filter algorithm, if there is more than one part in the antecedent (*IF part*) of the rules, a fuzzy operator must be used to combine the degrees of the input (X) and $\|X\|^2$ into a single value. Two commonly used fuzzy operators, *AND* and *OR* to combine the 2 variables are examined. It has been found that the *AND* operator, which chooses the *MIN* tends to have better result than the *OR* operator. This is followed by applying the implication method that is defined on the shaping of the consequent (*THEN-part*) of the rule based on the antecedent. In this case, a *min* (minimum) operation that truncates the output fuzzy set for each rule is preferred.

5.3.5 Aggregation and Defuzzification Process

The next step in the fuzzy inference engine is to aggregate all the outputs of each rule into a single fuzzy set for the adaptive gain variable. The final process of the FIS is to convert or defuzzify the aggregated fuzzy value for the adaptation gain into a crisp value to be used by the filter parameter vector updated law (5.1). There are many Defuzzification methods [48] available and the following centroid calculation that returns the centre of area under the aggregated *MBF*s curve is being employed here:

$$g(k) = \frac{\sum_{i=1}^{j} g_k(i) F(g_k(i))}{\sum_{i=1}^{j} F(g_k(i))} \tag{5.5}$$

where j is the number of sections used in approximating the area under the aggregated *MBF* and $F(g_k(i))$ is the *MBF* value at location, $g_k(i)$. The reason for using the centroid method instead of other defuzzification methods [48] such as the bisector, the middel of maximum (*mom*), the smallest of maximum (*som*) and the largest of maximum (*lom*), is because the centroid method produces the smallest mean square error and lends itself well to implementing on DSP. The other approaches require comparison operations to be carried out which complicate the implementation of defuzzification in DSP.

5.4 Simulation Examples: Fuzzy Gain Lyapunov Filter

In this section, some preliminary simulation results of the proposed fuzzy gain Lyapunov adaptive filter are presented here. For a comparative study, the adaptive numerical filters with RLS and LAF algorithms are also accomplished. These results are intended to show the proposed fuzzy gain Lyapunov adaptive filter can have comparable performance to the numerical filters and also allow for the simple inclusion of heuristic knowledge.

Example 1- Fuzzy Gain Filter

The desired signal $d(k)$ and the filter input signal $x(k)$ are shown in *Figure 5.5*. The additive noise, $n(k)$ is a bounded random noise which satisfies the following bounded condition: $|n(k)| \leq 0.4$. The filter parameters are adaptively updated by the crisp output value of defuzzification in the expression (5.5). *Figure 5.6* has revealed the performance of the fuzzy gain filter.

Example 2 -Numerical filters

For the same setup, the performance of the numerical filters with the LAF and RLS algorithms are revealed. *Figure 5.7* and *Figure 5.8* show the LAF filter output when the smaller and larger λ_1, λ_2 and κ parameters in the expression (5.4) are used respectively. It has been shown in Chapter 3 that the performance of the filter depends on the parameters, λ_1, λ_2, κ. On the other hand, *Figure 5.9* and *Figure 5.10* reveal the performance of the RLS filter depending on the forgetting factor, ρ critically. The small forgetting factor gives good filtering performance but the adaptive parameters tend to vary in very large magnitude.

Mean Square error plots for example 1 and example 2:

For comparison, the mean square error (MSE) from the iteration 1 to 1000 for the simulation examples 1 and 2 are plotted. Figures 5.11, 5.12, 5.13, 5.14 and 5.15 are the MSE plots of the Figures 5.6, 5.7, 5.8, 5.9 and 5.10 respectively.

From the results, it is observed that the fuzzy gain filter without exact mathematical model can give equivalent performance as the numerical filters provided the fuzzy rules and regions of membership function are designed properly. The fuzzy gain filter can deal with many ambiguous or uncertain situations and the exact mathematical model is not required. User can also extrapolate MBFs and rules manually from their experience to suit different applications.

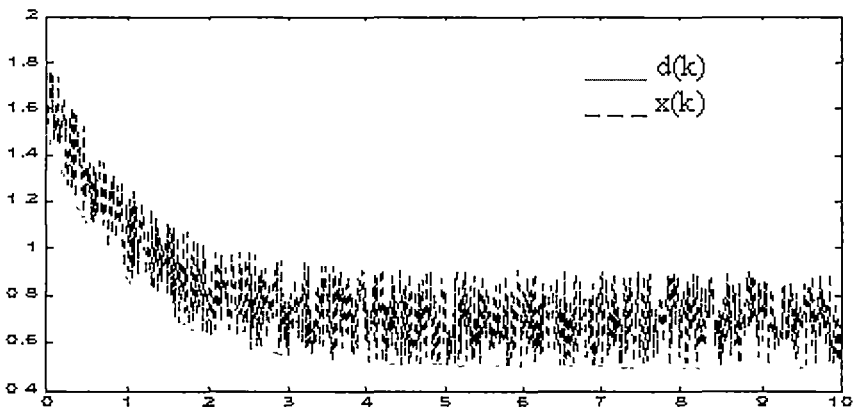


Figure 5.5: The desired signal *d(k)* and the filter input signal *x(k)*

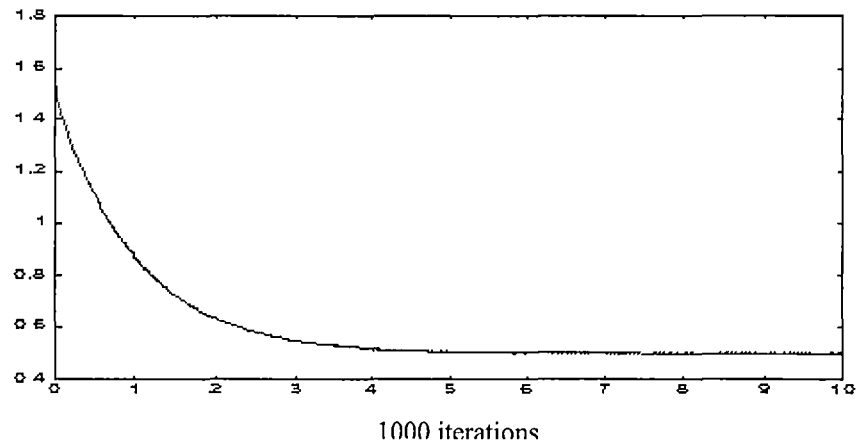


Figure 5.6: Fuzzy Gain Lyapunov filter - output signal, *y(k)*

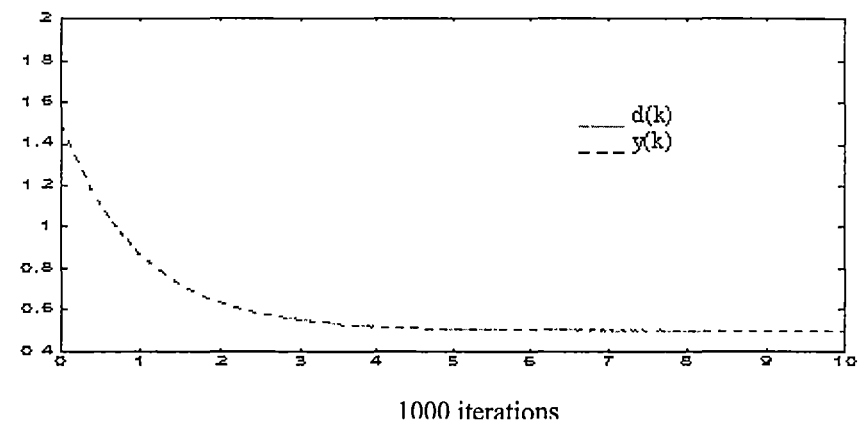


Figure 5.7: LAF (small λ_1,λ_2, $\kappa = 0.01$) - the desired signal *d(k)* and filter output *y(k)*

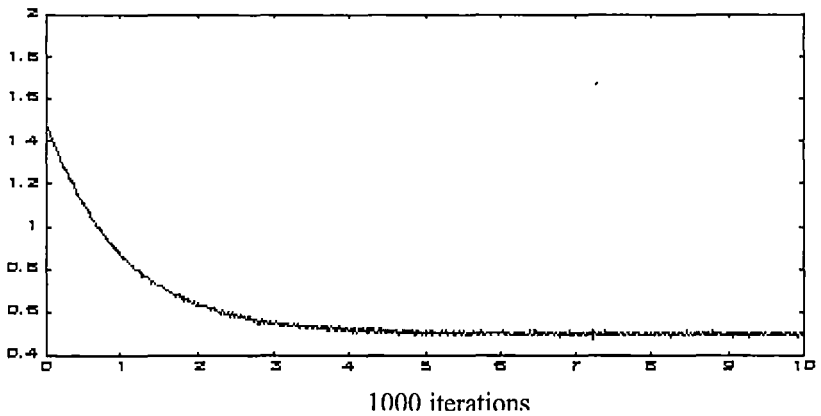


Figure 5.8: LAF (large λ_1,λ_2, $\kappa = 0.4$) - the desired signal *d(k)* and filter output *y(k)*

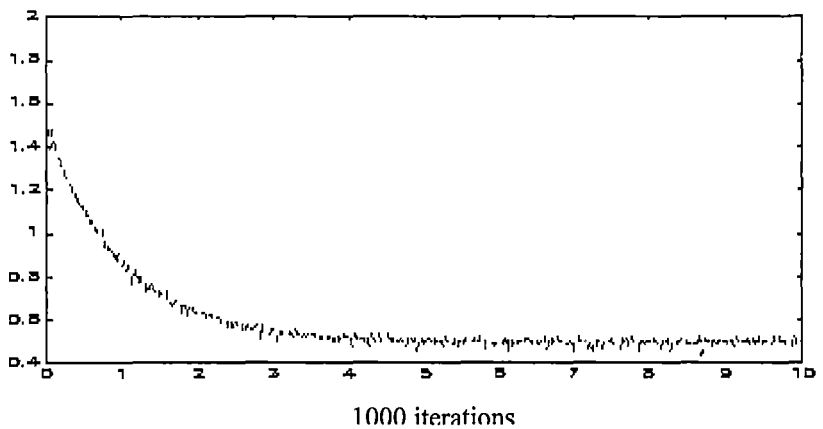


Figure 5.9: RLS (large ρ=0.6) - the desired signal *d(k)* and filter output *y(k)*

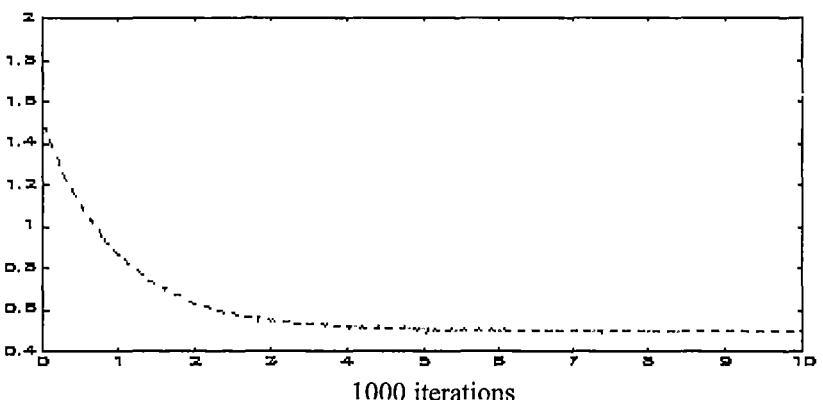


Figure 5.10: RLS (small ρ=0.1) - the desired signal *d(k)* and filter output *y(k)*

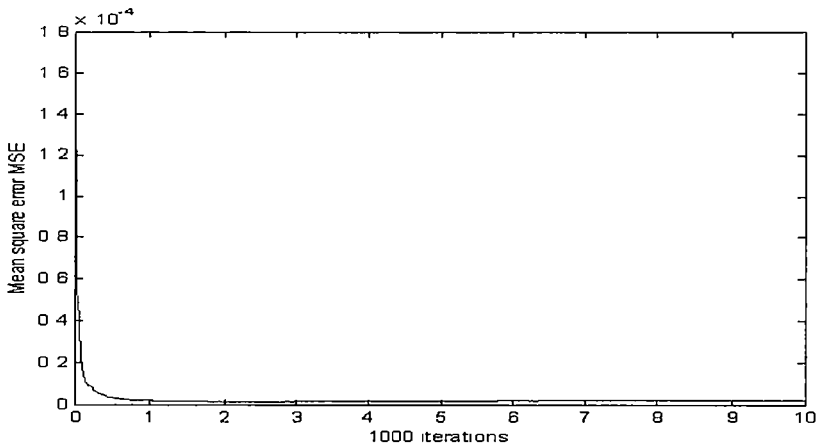


Figure 5.11: MSE plot of Figure 5.6 (*y-axis: x10^{-4}*)

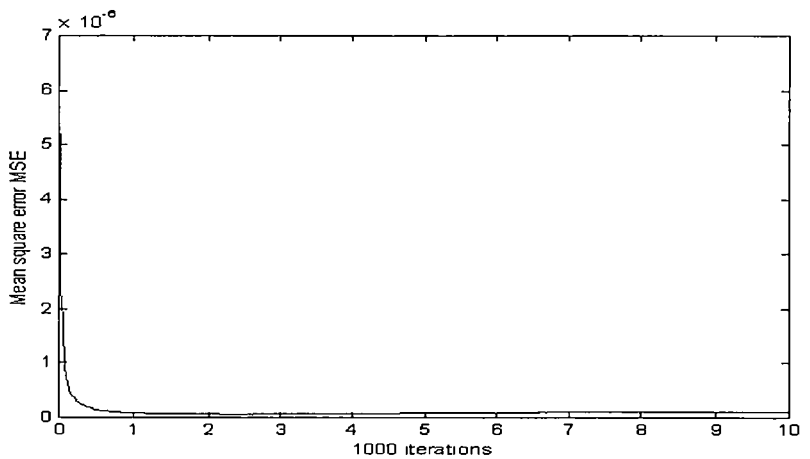


Figure 5.12: MSE plot of Figure 5.7 (*y-axis: x10^{-6}*)

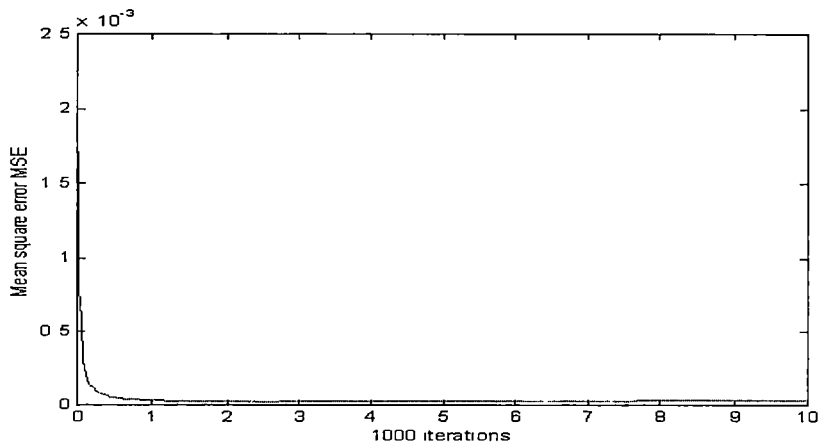


Figure 5.13: MSE plot of Figure 5.8 (*y-axis: x10^{-3}*)

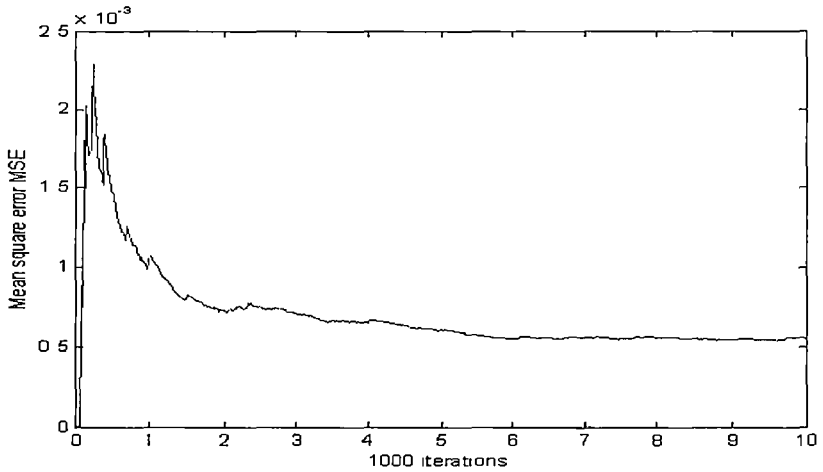


Figure 5.14: MSE plot of Figure 5.9 (*y-axis: $x10^{-3}$*)

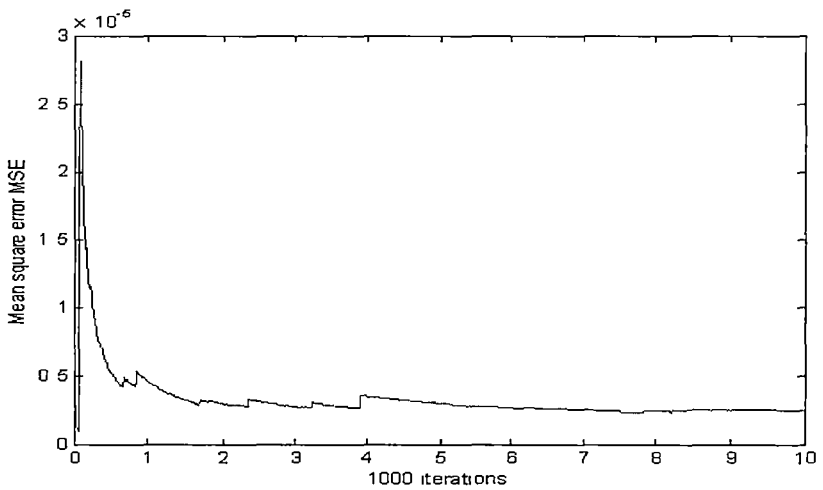


Figure 5.15: MSE plot of Figure 5.10 (*y-axis: $x10^{-5}$*)

5.5 LAF Fuzzy Adaptive Filter

In this section, another new fuzzy adaptive filter using Lyapunov stability theory is proposed. This fuzzy adaptive filter is constructed from a set of changeable fuzzy IF-THEN rules. The adaptive algorithm, *Lyapunov theory-based adaptive filtering* (LAF) is used to update the parameter of the membership functions so that the dynamic error between the filter output and the desired response converges to zero asymptotically.

Our LAF fuzzy adaptive filter solves the following problem. Consider a real-valued input sequence $[x(k)]$ and a real-valued desired output sequence $[d(k)]$, where $k = 0, 1, 2, \ldots$ is the time index. At each time point k, we are given the values of $x(k)$ and $d(k)$. The problem is to determine an adaptive filter $f(x(k))$ such that the dynamic error can converge to zero asymptotically.

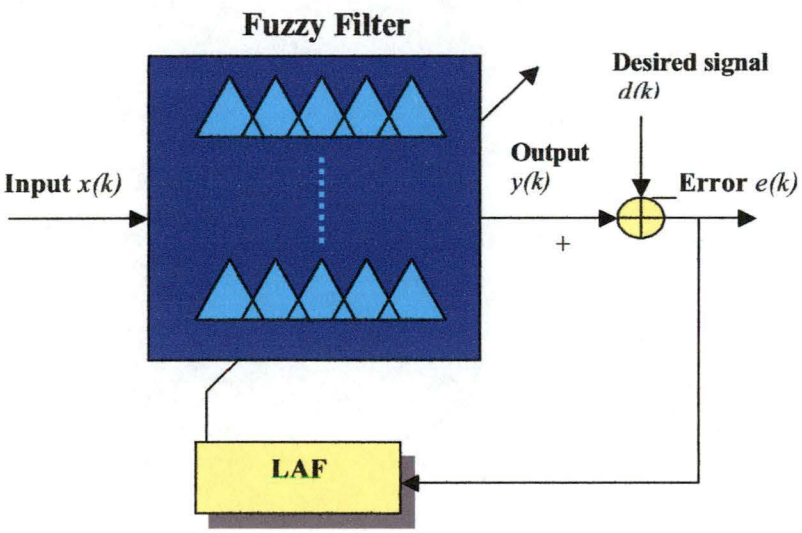


Figure 5.11: The LAF Fuzzy Filter

There are several approaches in fuzzy adaptive filters [45],[47],[1],[49],[50]. Authors of [45] have presented a fuzzy adaptive filter that is constructed from a set of changeable fuzzy IF-THEN rules to minimize some criterion functions. These fuzzy adaptive filters parameters are updated by RLS and LMS algorithms. However, the computation complexity involved in RLS fuzzy filter is highly parallelizable and the

fuzzy RLS filter might not be able to be used in some practical situations where the computing power is limited [45]. Therefore another fuzzy adaptive filter that involves much less computation is essential. On the contrary, the LMS fuzzy filter [45] has suffered from slow error convergence.

Before we discuss the design detail of the LAF fuzzy adaptive filter, we first give a brief summary of the procedure. The Lyapunov fuzzy adaptive filter is constructed through the following steps. First, fuzzy sets are defined in the filter input space $X \subset R^n$ whose membership functions cover X. Then a set of fuzzy IF-THEN rules which either come from human experts or are determined during the adaptation procedure by matching input-output data pairs is constructed. A filter based on this set of rules is constructed and its free parameters are updated using the LAF algorithm. The design procedure is similar to that in [45] that the fuzzy adaptive filter using RLS algorithm. This scheme has less computation complexity than the RLS fuzzy filter in [45]. The stability of the fuzzy filter is guaranteed by Lyapunov stability theory.

5.6 Design Procedure of the LAF Fuzzy Adaptive Filter

The design procedure of the LAF fuzzy adaptive filter is listed as follow:

Step 1:

Define M fuzzy sets F_i^l in each interval $[C_i^-, C_i^+]$ of the input space U. The M membership functions $\mu_{F_i^l}$ cover the interval $[C_i^-, C_i^+]$ and $\mu_{F_i^l}$ s are fixed functions. For exmaple, Gaussian membership functions

$$\mu_{F_i^l}(x_i) = \exp\left[-\frac{1}{2}\left(\frac{x_i - \bar{x}_i}{\sigma_i^l}\right)^2\right] \tag{5.6}$$

where $l = 1,2,\ldots, M$, $i = 1,2,\ldots,n$, $x_i \in [C_i^-, C_i^+]$, σ_i^l and $\bar{x}_i$ are fixed parameters.

Step 2:

Construct a set of M fuzzy IF-THEN rules in the following form:

$$R^l: \text{IF } x_1 \text{ is } F_1^l \text{ and } \ldots \text{ and } x_n \text{ is } F_n^l, \text{ THEN } y \text{ is } G^l \tag{5.7}$$

where $\tilde{x} = (x_1, \ldots, x_n)^T \in U$, $y \in R$, F_i^l's are defined in Step 1, and G^l's are fuzzy sets defined in R which are determined as follows: if there are linguistic rules in the form of (5.7), set F_i^l's and G^l to be those labels of these linguistic rules; otherwise, choose parameter μ_{G^l} arbitrarily. The (parameter of) membership functions μ_{G^l} in these rules will change during the LAF adaptation procedure of Step 4. Therefore the rules constructed in this step are initial rules of the fuzzy adaptive filter. We incorporate linguistic rules into the LAF fuzzy adaptive filter by constructing the initial filter based on these rules.

Step 3:

Construct the filter f_k: $U \rightarrow R$ based on the M rules of Step 2 as follows:

$$f_k(\tilde{x}) = \frac{\sum_{l=1}^{M} \theta^l \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i) \right)}{\sum_{l=1}^{M} \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i) \right)} \tag{5.8}$$

where $\tilde{x} = (x, \ldots, x_n)^T \in U$, $\mu_{F_i^l}$'s are the membership functions of filter input, eg., Gaussian membership function of (5.6), and $\theta^l \in R$ is any point at which μ_{G^l} achieves its maximum value. If we chose the membership functions $\mu_{F_i^l}(x_i)$ to be Gaussian functions which are nonzero for any $x_i \in [C_i^-, C_i^+]$, the denominator of (5.8) is nonzero for any $\tilde{x} \in U$. Therefore, the filter f_k of (5.8) is well defined. In (5.8), the weights $\mu_{F_i^l}$ of the fuzzy adaptive filter are fixed functions. Therefore, the free design parameters of the fuzzy adaptive filter are the θ^l. We can now rewrite (5.8) as

$$f_k(\tilde{x}) = p^T(\tilde{x})\theta^* \tag{5.9}$$

where $\theta^* = [\theta^1, \theta^2, \ldots, \theta^M]^T$, $p = [p_{l=1}, p_{l=2}, \ldots, p_{l=M}]^T$ and

$$p_l = \left(\mu_{F_1^l}(x_1).\mu_{F_2^l}(x_2)\ldots\mu_{F_n^l}(x_n) \right) \div \sum_{l=1}^{M} \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i) \right)$$

Step 4:

Use the LAF algorithm to update the filter parameters, θ^l so that the error can converge to zero asymptotically. Let the initial *eg.* $\theta^l(0)$ be determined in Step 2; at each time point $k = 1,2,\ldots$ do the following:

$$\theta^*(k) = \theta^*(k-1) + g(k)\alpha(k) \tag{5.10}$$

where $g(k)$ is the adaptation gain and $\alpha(k)$ is a priori estimation error defined as

$$\alpha(k) = d(k) - \theta^*(k-1)p(k) \tag{5.11}$$

The adaptation gain $g(k)$ in (5.10) is adaptively adjusted using Lyapunov stability theory as (5.12) so that the error $e(k)$ asymptotically converges to zero.

$$g(k) = \frac{p(k)}{\| p(k) \|^2}\left(1 - \kappa\frac{|e(k-1)|}{|\alpha(k)|}\right) \tag{5.12}$$

where $0 \le \kappa < 1$. The deficiencies of the expression (5.12) that the values of $p(k)$ and $\alpha(k)$ may be zero and rise singularity problem are also noticed. Therefore the adaptation gain may be modified as the adaptation law (5.13) to avoid singularities.

$$g(k) = \frac{p(k)}{\lambda_1 + \| p(k) \|^2}\left(1 - \kappa\frac{|e(k-1)|}{\lambda_2 + |\alpha(k)|}\right) \tag{5.13}$$

where λ_1, λ_2 are small positive numbers.

The following remarks are some comments on the LAF fuzzy filter:

Remark 5.1: The LAF algorithm (5.10)-(5.13) is obtained by modifying the RLS using Lyapunov theory. Because f_k of (5.9) is linear in the parameter, the derivation of (5.10)-(5.13) is the same as that of the FIR adaptive filter in [18]. Therefore we omit the details.

Remark 5.2: The LAF algorithm can be viewed as updated the rules in the form of (5.7) by changing the 'centers', θ^l of the THEN parts of these rules so that the error can converge to zero asymptotically. We are allowed only to change these 'centers'. The membership functions $\mu_{F_i^l}$ of the IF parts of the fules are fixed at the very

beginning and are not allowed to change. Thus a good choice of $\mu_{F_i^l}$'s is important to the success of the entire filter.

Remark 5.3: It was proven in [46] that functions in the form of (5.8) are universal approximators. That is, for any real continuous function q on the compact set U, there exists a function in the form of (5.8) such that it can uniformly approximate q over U to arbitrary accuracy. Consequently, the fuzzy adaptive filter is a powerful nonlinear adaptive filter in the sense that it has the capability of performing difficult nonlinear filtering operations.

Remark 5.4: The fuzzy adaptive filter (5.9) performs a two-stage operation on the input, *x(k)*. First, it performs a nonlinear transformation *p(.)* on *x(k)*; then the filter output is obtained as a linear combination of these transformed signals. In this sense, the fuzzy adaptive filter is similar to the radial basis function [51],[52] approaches. However, the unique feature of the fuzzy filter, which is not shared by other nonlinear adaptive filters, is that linguistic rules can be incorporated into the filter.

Remark 5.5: Linguistic information (in the form of the fuzzy IF-THEN rules of (5.7)) and numerical information (in the form of desired input-output pairs (*x(k)*, *d(k)*)) are combined into the filter in the following way. Due to Steps 2-4, linguistic IF-THEN rules are directly incorporated into the filter (5.8) by constructing the initial filter based on the linguistic rules.

5.7 Simulation Examples: LAF Fuzzy Adaptive Filter

In this section, some preliminary simulation results of the LAF fuzzy adaptive filter are presented. The advantage of the proposed filters is that linguistic information from human experts (in the form of fuzzy IF-THEN rules) can be incorporated into the filters. Detail of the linguistic information will not be included here. The LAF fuzzy adaptive filter can deal with many ambiguous or uncertain situations and the exact mathematical model is not required. A bounded additive noise: $0 < n(k) < 0.2$ is introduced at the filter input.. *Figure 5.12* illustrates the comparison of the desired signal, *d(k)* and the filter output, *y(k)* respectively (2000 samples). The error, *e(k)* which is the difference between *d(k)* and *y(k)* is shown in *Figure 5.13*.

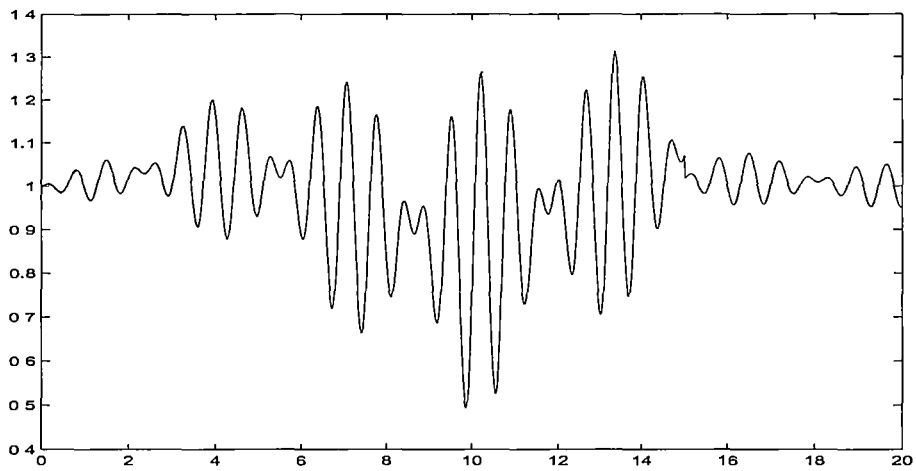


Figure 5.12: LAF fuzzy filter - the desired signal, *d(k)* & filter output, *y(k)*

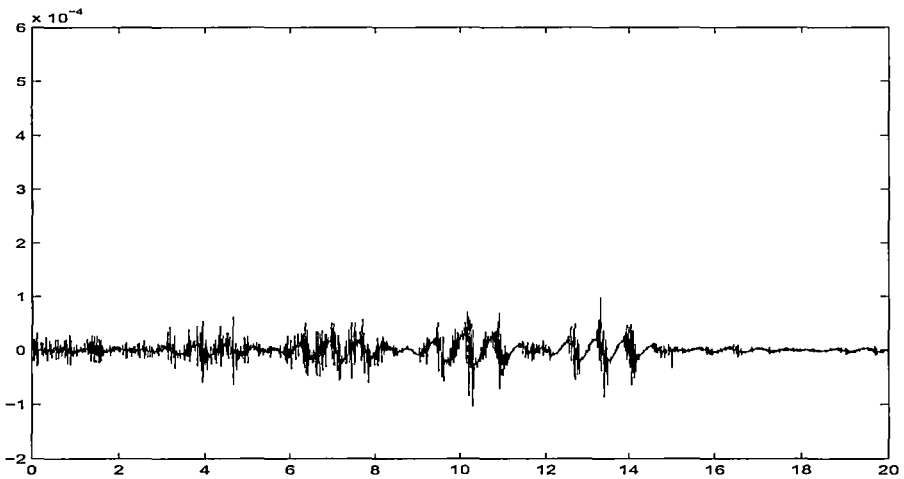
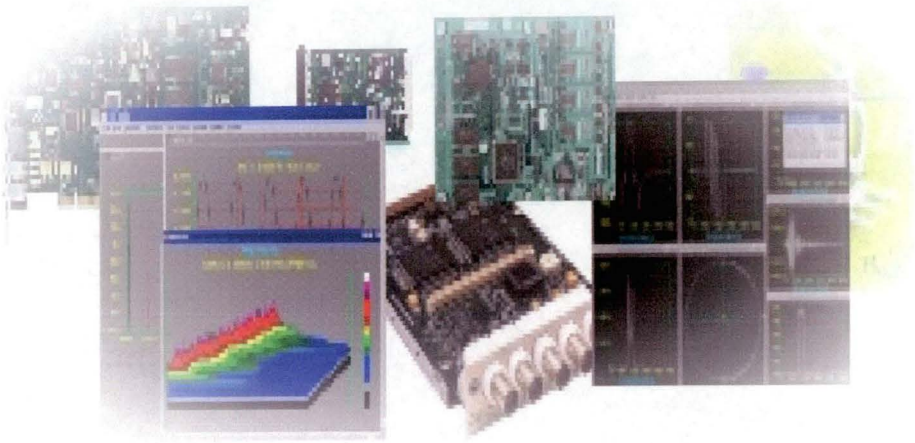


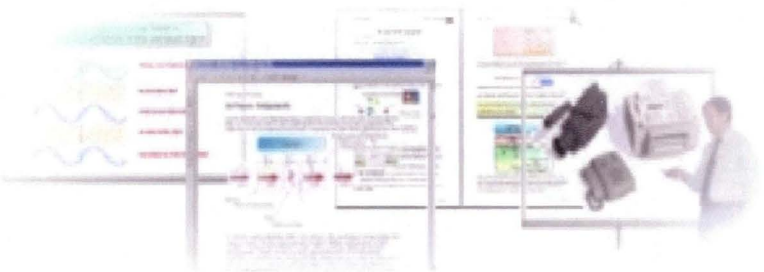
Figure 5.13: LAF fuzzy filter - the error, *e(k)* (*y-axis:* $\times 10^{-4}$)

5.8 Conclusion

In this chapter, we have developed two nonlinear adaptive filters based on the *Lyapunov theory-based adaptive filtering* (LAF), namely, *the fuzzy gain Lyapunov adaptive filter* and *LAF fuzzy adaptive filter*. The developed adaptive fuzzy gain filters utilize both numerical data and linguistic information expressed by fuzzy IF THEN rules. The IF THEN rules are designed based on the Lyapunov theory. Hence the fuzzy gain filter with Lyapunov sense fuzzy rules can lead to error convergence to zero asymptotically. Furthermore it is possible to incorporate other *a priori* knowledge into the filter design. On the other hand, the key elements of the LAF fuzzy adaptive filter is the fuzzy logic system, which is constructed from a set of fuzzy IF-THEN rules, and the LAF adaptive algorithm for updating the parameters in the fuzzy system. The parameters in the fuzzy system are adjusted adaptively so that the error convergence to zero asymptotically. The most significant advantage of the fuzzy adaptive filters is that linguistic information from human experts (in the form of fuzzy IF-THEN rules) can be incorporated into the filters. If no linguistic information is available, the fuzzy adaptive filters become well-defined nonlinear adaptive filters, similar to the polynomial, neural networks, or radial basis function adaptive filters. The simulation examples have verified the aforementioned theoretical analyses of both fuzzy filters.

Chapter 6





Lyapunov Stability-Based Adaptive Backpropagation (LABP) For Discrete-time Dynamical System

Chapter 6

Lyapunov Stability-Based Adaptive Backpropagation (LABP) For Discrete-time Dynamical System

6.1 Introduction

Recently dynamic neural networks have been attracting much attention from scientific community (special issue of 1994 March, IEEE Transactions of Neural Networks and 1997 June, Neurocomputing) because they are useful for temporal processing such as digital signal processing (DSP), system identification and control. There are two main methods to provide a static neural network with dynamic behavior: the insertion of a buffer somewhere in the network, or the use of feedback. The first kind of dynamic network is a *buffered multilayer perceptron* (MLP) in which tapped delay lines (TDLs) of the input are used. The buffer can be applied at the network inputs only, keeping the network internally static as in buffered MLP's [53] (*Figure 6.1a*), or at the input of each neuron as in MLP with FIR filter synapses [44] (*Figure 6.1b*). The structure in *Figure 6.1a* is often called *time-delay neural network* (TDNN) in [42],[55] and *adaptive time-delay neural networks* in [56]-[57]. It is well known that the buffered MLP and FIR-MLP can be shown to be theoretically equivalent [54] since internal buffers can be implemented as an external one. The problem with implementing FIR-MLP's as buffered MLP's is that the first layers sub networks must be replicated (with shared weights) [43] and so the complexity is much higher. Therefore buffered MLP and FIR-MLP are different

architectures with regard to a real implementation. In this chapter, we have merely considered the TDNN or the buffered MLP with TDLs.

Examples of implementation of feedback in the recurrent neural networks are in [28],[44],[42],[58]-[60],[43],[27],[61]-[65]. The major difference among these methods lies in how the feedback is included in the network. The feedback can be included externally as the *Narendra-Parthasarathy* MLP [66] also know as NARX network, where TDL's are used for the outputs that feedback to the input of the network. (*Figure 6.1c*), and in the *Elman's* network [59]. If the feedback is connected internally or inside each neuron, this approach is called *locally recurrent neural networks* (LRNN's) or *local feedback multilayer network* (LF_MLN) [43],[60]. In these structures, classical IIR linear filters [18], also called ARMA models are used either directly or with some modifications. (*Figure 6.1b* with IIR structure). The formal structure with external feedback is considered in this chapter.

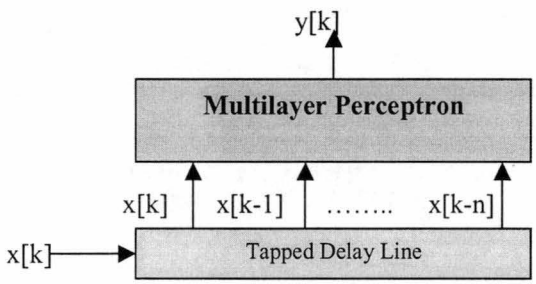


Figure 6.1a: TDNN or MLP with TDL's inputs

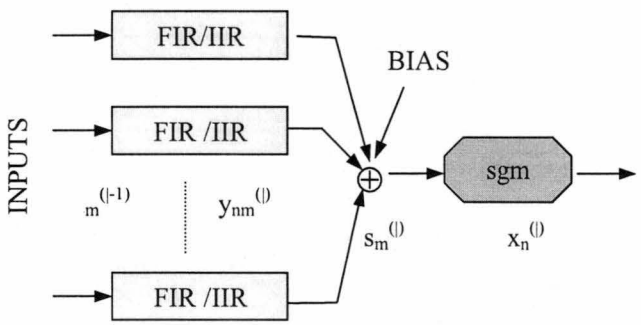


Figure 6.1b: FIR-MLP or IIR-MLP Local Recurrent Neural Network

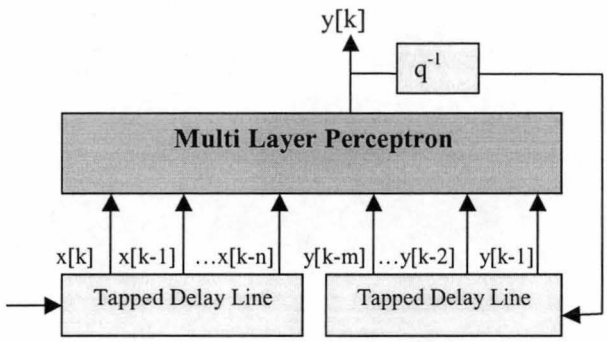


Figure 6.1c: MLP with TDL's inputs and outputs, sometimes called Narendara Parthasarathy or NARX neural network.

A number of gradient-based algorithms have been developed for adapting discrete-time dynamical systems [67],[42]. For the buffered MLP of *Figure 6.1a* with only input buffer, the *Backpraopagation* (BP) can be used. BP is probably the most widely applied neural network learning algorithm. Extension of BP to recurrent networks were proposed in [6]-[7],[13],[16]. Two main gradient-based learning approaches exist for recurrent networks: *Backpropagation through time* (BPTT) [28],[1],[42],[57] and *real-time recurrent learning* (RTRL) [40],[38],[57]. The difference between BPTT and RTRL is in how the chain rule derivative expansion is applied. More specifically during the learning phase, in BPTT the neural network is computed backward both in the layer and time dimensions, whereas in RTRL it is calculated forward.

As pointed out by numerous researchers, BP or its modifications may suffer from slow convergence and may be trapped in local minima during gradient descent [70],[68]. There are different optimization numerical attempts [68]-[70],[38] to solve these problems. Unfortunately, as for any nonlinear optimization problem, we do not have 'a priori', guarantees that the numerical solving scheme will approach the optimal solution. The main difficulty is with the 'intrinsic shape' of the cost surface which is normally fixed and independent of the learning algorithm. As a result any algorithm must deal with such a surface. Therefore there are likely to be 'easy and difficult' problems, depending on the shape of that surface and not on the learning algorithm. This does not mean that no learning procedure can effectively find optimal solution. However, if the cost function has many local minima, devising an

effective learning algorithm may be very difficult or may involve high computational burden. Furthermore, the stability of the weight updated algorithm itself is also a significant problem when training dynamic neural networks [28].

To overcome the above problems, a new approach of designing a BP algorithm using Lyapunov stability theory is proposed in this chapter. This chapter has also extended the ideals of *Lyapunov stability-based* algorithms in Chapter 3-5 to the design of the BP algorithm for TDNN with feedback in particular. We call this new algorithm as *Lyapunov Stability-based Adaptive Backpropagation* (LABP) algorithm. The designed LABP is a non-gradient based algorithm. In our new scheme, a Lyapunov function is defined for the error between the desired response and the neural network output. The defined criterion function is the Lyapunov function that has only unique minimum. The weights of neural network are then adaptively adjusted so that the error can converge to zero asymptotically. The network weights updated strategy is independent of signal statistical properties because only the desired response and input signal are required. Its computational complexity is less than the algorithms such as genetic algorithms, learning automata and simulated annealing [33]-[35]. The stability concern for the LABP algorithm is guaranteed by the Lyapunov Stability Theory. Simulation examples are included to demonstrate the good performance of the proposed scheme.

This chapter is organized as follows. In section 6.2, the LABP for TDNN is presented. Section 6.3 discusses the design of the proposed LABP algorithm. Section 6.4 reveals the extension of the LABP algorithm to the recurrent MLP with TDLs of inputs and feedback outputs. The theoretical derivation is further supported by the simulation examples in section 6.5. Finally, section 6.6 concludes this chapter.

6.2 Lyapunov Stability-based Adaptive Backpropagation (LABP) for Buffered MLP with TDL's Inputs or TDNN

Before we discuss our new LABP algorithm for buffered MLP with TDL's inputs, we first discuss the *Lyapunov theory-based adaptive algorithms* proposed in Chapter 3-5. The Lyapunov stability-based adaptive algorithms used in Chapter 3-5 are the modification of re*cursive least squares* (RLS) algorithm using Lyapunov stability theory. They are different from the gradient-based methods in the optimization techniques. The selected Lyapunov function for the design has a unique global minimum in the state space. By properly choosing the parameter or weight update law in Lyapunov sense, the output of the adaptive filter or RBF network can asymptotically converge to the desired reference signal. The so-called local minima problem occurred in the gradient search-based algorithm can be prevented. The design is independent of the stochastic properties of the input disturbances since only the input observations and a collection of desired response are required. These Lyapunov stability-based algorithms have provided new approaches in adaptive filtering and RBF neural filtering designs. They may give alternative solution to the problems encountered in gradient-based methods such as standard BP for the TDNN. Therefore, we extend the Lyapunov theory-based ideals to the LABP for TDNN in this section.

6.2.1 Architecture

The architecture of the feedforward dynamical MLP we consider for training is shown in *Figure 6.2*. The input $x(k)$ is a sampled signal: $x(k) = \{x(k), x(k-1),\ldots x(k-n)\}$ or $x(k) = \{x_k, x_{k-1}, \ldots, x_{k-n}\}$. The output is a scalar $y(k)$. The purpose of this neural network is to adjust the neural weights $\{W_{j,i}^{(l,\ l-1)}(k)\}$ in order to achieve error between the network output $y(k)$ and the desired output $d(k)$ converge to zero asymptotically. l is the layer index, j the node or neuron index, i the connection index, $S_j^{(l)}(k)$the output node j in the layer l, and $W_{j,i}^{(l,\ l-1)}$ the ith weight related to the j node in the layer l with respective to the previous layer, l-1. $N(l)$ denotes the number of neurons in layer l. All the neural network activation functions is sigmoid

function, thus the output of the *j*th node of the *l*th layer is a function of the weighted sum of the outputs of the preceding layer. (*Figure 6.2*, note that only one intermediate layer has been drawn)

$$S_j^{(l)}(k) = F_j\left(\sum_{i=1}^{N(l)} W_{ji}^{(l,l-1)}(k) S_i^{(l-1)}(k)\right) \tag{6.0}$$

The network nodes are partitioned into layers measured *0* to *L+1*, where the layer number indicates the distance of a node from the input nodes. The lower most layer is the input layer numbered as layer *0*, and the topmost layer is the output layer numbered as layer *L+1*. BP addresses networks for which $L \geq 1$, containing "hidden layers" numbered *1* to *L*. For convenience of presentation, we will assume that $L = 1$ in describing the LABP algorithm, implying that there is only one hidden layer, as shown in *Figure 6.2*. The algorithm can be extended easily to the cases when $L \neq 1$. There are N "real" inputs, thus the *i*th real input contribution for the MLP inputs is $x_i(k) = \{x_i(k), x_i(k-1), \ldots x_i(k-n)\}$ with $i=1,2 \ldots N$.

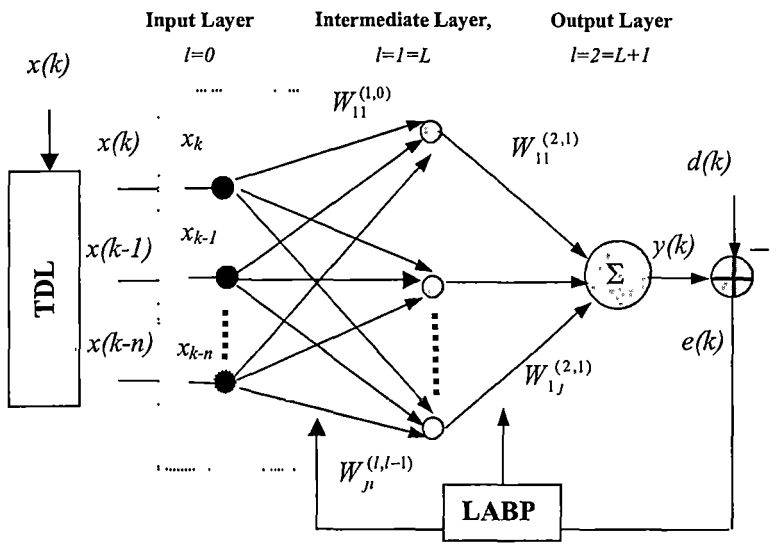


Figure 6.2: TDNN or MLP with TDL's Inputs

The following is the scenario for the feedforward network with $L = 1$ and with N=1. Let $W_{ji}^{(l,0)}(k)$ denote the connection weights between the *i*'th neuron in the input layer, *l=0* and *j*'th neuron in the hidden layer, *l=1* ($j = 1, 2, \ldots, N(l=1)$ and $i = 1, 2,$

..., $N(l=0)$), where $N(l=1)$ is the number of nodes in the layer $l = 1$ and $N(l=0)$ is the number of nodes in the layer $l = 0$. Let $S_j(k)$ and $F_j(\cdot)$ be the output and the activation function of the j'th neuron in the hidden layer, respectively. $W_{1j}^{(2,1)}(k)$ denote the connection weights between the j'th neuron in the hidden layer and the neuron in the output layer $y(k)$, $l=2$. Then we have

$$y(k) = \sum_{j=1}^{N(l=1)} W_{1j}^{(2,1)} S_j(k) \tag{6.1}$$

$$S_j(k) = F_j\left(\sum_{i=1}^{N(l=0)} W_{ji}^{(1,0)} x_i(k) \right) \tag{6.2}$$

where $j = 1, 2, ..., N(l=1)$ and $i = 1, 2, ..., N(l=0)$

Substituting (6.2) into (6.1) gives

$$y(k) = \sum_{j=1}^{N(l=1)} W_{1j}^{(2,1)} F_j\left(\sum_{i=1}^{N(l=0)} W_{ji}^{(1,0)} x_i(k) \right) \tag{6.3}$$

where $F(\bullet) = \dfrac{1}{1+e^{-\alpha(\bullet)}}$

Note: the subscript of $W_{1j}^{(2,1)}(k)$ is 1 (one) followed by j. '1' indicates the output layer has only one output node.

6.2.2 LABP Training Algorithm

The strategy for updating the network parameters involves adaptive and supervised learning. At each iteration LABP algorithm is used to update the weights of the neural network using Lyapunov theory [20]. The connection weights $W_{ji}^{(1,0)}(k)$ and $W_{1j}^{(2,1)}(k)$ are first initialized randomly. Then the input vector $x_i(k)$ is passed successively to the input layer of the feedforward neural network. The output $S_j(k)$) of the hidden layer and the output $y(k)$ are computed using (6.1)-(6.3) and the available input vector $x_i(k)$. This is followed by computing the error, $e(k)$.

$$e(k) = y(k) - d(k) \tag{6.4}$$

The weights of the TDNN can be updated using the following expressions:

$$W_{1J}^{(2,1)}(k) = W_{1J}^{(2,1)}(k-1) + \Delta W_{1J}^{(2,1)}(k-1) \tag{6.5}$$

and

$$W_{JI}^{(1,0)}(k) = W_{JI}^{(1,0)}(k-1) + \Delta W_{JI}^{(1,0)}(k-1) \tag{6.6}$$

where

$$\Delta W_{1J}^{(2,1)}(k) = \frac{1}{S_J(k)} \frac{1}{N(l=1)} \left[d(k) - \sum_{J=1}^{N(l=1)} W_{1J}^{(2,1)} S_J(k) \right] \tag{6.7}$$

$$\Delta W_{JI}^{(1,0)}(k) = \left[-W_{JI}^{(1,0)}(k-1) + \frac{1}{N(l=0)} \frac{1}{x_I(k)} g_J(u(k)) \right] \tag{6.8}$$

$$u(k) = \frac{1}{N(l=1)} \frac{1}{W_{1J}(k)} d(k) \tag{6.9}$$

$$g_J(\bullet) = F_J^{-1}(\bullet) \tag{6.10}$$

The design detail is presented in the next section.

Remark 6.1: According to [71], the sigmoidal activation function is *one-to-one* and this is also said to be invertible. Therefore, $g_J(\bullet)$ has the inverse function, thus the inverse function, $F_J^{-1}(\bullet)$ exists.

6.3 Design of LABP Using Lyapunov Theory

The design of the LABP for the TDNN or the MLP with TDL of inputs is described by Theorem 6.3.1:

Theorem 6.3.1: For the given input $x_I(k)$, if the weights $W_{J,I}^{(1,0)}(k)$ and $W_{1J}^{(2,1)}(k)$ of the neural network are updated as follows

$$W_{1J}^{(2,1)}(k) = W_{1J}^{(2,1)}(k-1) + \Delta W_{1J}^{(2,1)}(k) \tag{6.11}$$

$$\Delta W_{1J}^{(2,1)}(k) = \frac{1}{S_J(k)} \frac{1}{N(l=1)} \left[d(k) - \sum_{j=1}^{N(l=1)} W_{1J}^{(2,1)} S_j(k) \right] + \alpha e(k-1) \qquad (6.12)$$

and $$W_{JI}^{(1,0)}(k) = W_{JI}^{(1,0)}(k-1) + \Delta W_{JI}^{(1,0)}(k) \qquad (6.13)$$

$$\Delta W_{JI}^{(1,0)}(k) = \left[-W_{JI}^{(1,0)}(k-1) + \frac{1}{N(l=0)} \frac{1}{x_I(k)} g_J(u(k)) \right] \qquad (6.14)$$

where $$u(k) = \frac{1}{N(l=1)} \frac{1}{W_{1J}(k)} d(k) \qquad (6.15)$$

$$g_J(\bullet) = F_J^{-1}(\bullet) \text{ and } 0 < \alpha < 1 \qquad (6.16)$$

then the error *e(k)* in (6.4) converges to zero asymptotically.

Proof: Define a Lyapunov function of error *e(k)*

$$V(k) = e^2(k) \qquad (6.17)$$

$$\Delta V(k) = V(k) - V(k-1)$$

$$= e^2(k) - e^2(k-1)$$

$$= (y(k) - d(k))^2 - e^2(k-1)$$

$$= \left(\sum_{j=1}^{N(l=1)} W_{1J}^{(2,1)}(k) S_j(k) - d(k) \right)^2 - e^2(k-1)$$

$$= \left(\sum_{j=1}^{N(l=1)} \left(W_{1J}^{(2,1)}(k-1) + \Delta W_{1J}^{(2,1)}(k) \right) S_j(k) - d(k) \right)^2 - e^2(k-1)$$

$$= \left(\sum_{j=1}^{N(l=1)} W_{1J}^{(2,1)}(k-1) S_j(k) + \sum_{j=1}^{N(l=1)} \Delta W_{1J}^{(2,1)}(k) S_j(k) - d(k) \right)^2 - e^2(k-1)$$

$$
\begin{aligned}
&= \left(\sum_{j=1}^{N(l=1)} W_{1j}^{(2,1)}(k-1) F_j \left(\sum_{i=1}^{N(l=0)} W_{ji}^{(1,0)}(k) x_i(k) \right) + \right. \\
&\left. \sum_{j=1}^{N(l=1)} \Delta W_{1j}^{(2,1)}(k) F_j \left(\sum_{i=1}^{N(l=0)} W_{ji}^{(1,0)}(k) x_i(k) \right) - d(k) \right)^2 - e^2(k-1) \\
&= \left(\sum_{j=1}^{N(l=1)} W_{1j}^{(2,1)}(k-1) F_j \left(\sum_{i=1}^{N(l=0)} \left(W_{ji}^{(1,0)}(k-1) + \Delta W_{ji}^{(1,0)}(k) \right) x_i(k) \right) + \right. \\
&\left. \sum_{j=1}^{N(l=1)} \Delta W_{1j}^{(2,1)}(k) F_j \left(\sum_{i=1}^{N(l=0)} \left(W_{ji}^{(1,0)}(k-1) + \Delta W_{ji}^{(1,0)}(k) \right) x_i(k) \right) - d(k) \right)^2 \\
&- e^2(k-1) \\
&= \left(\sum_{j=1}^{N(l=1)} W_{1j}^{(2,1)}(k-1) F_j \left(\sum_{i=1}^{N(l=0)} W_{ji}^{(1,0)}(k-1) x_i(k) + \sum_{i=1}^{N(l=0)} \Delta W_{ji}^{(1,0)}(k) x_i(k) \right) + \right. \\
&\left. \sum_{j=1}^{N(l=1)} \Delta W_{1j}^{(2,1)}(k) F_j \left(\sum_{i=1}^{N(l=0)} W_{ji}^{(1,0)}(k-1) x_i(k) + \sum_{i=1}^{N(l=0)} \Delta W_{ji}^{(1,0)}(k) x_i(k) \right) - d(k) \right)^2 \\
&- e^2(k-1)
\end{aligned}
\tag{6.18}
$$

Using the expressions (6.12) and (6.14) in the expression (6.18), we have

$$\Delta V(k) = -(1-\alpha^2)\, e^2(k-1) < 0 \tag{6.19}$$

Remark 6.2: It is well-known that BP can be trapped in local minima during gradient descent. There are different modifications or improvements to the BP [70]. One of them is the adoption of *momentum term* [54] that improves the convergence speed and helps the network from being trapped in a local minimum. Besides the variation of BP, there are more complicated attempts [70],[33]-[35] to solve this problem and find the global minimum. Unfortunately, however, as for any nonlinear optimization problem, we do not have 'a priori' guarantees that the numerical solving scheme will approach the optimal solution [68]. The main difficulty is with the 'intrinsic shape' of the cost surface which is normally fixed and independent of the learning algorithm. The LABP provide a new attempt to this so-called local minima problem.

It is different from the gradient search based methods. According to Lyapunov stability theory, the selected $V(k)$ is a Lyapunov function if and only is $\Delta V(k)$ is negative ($\Delta V(k) < 0$). Whether or not $\Delta V(k)$ is negative depends on the selection of the weight updated laws. Only if the weight updated laws of the LABP is chosen in Lyapunov sense, then $V(k)$, the Lyapunov function has a unique global minimum. Therefore, the selection of the Lyapunov function and the weight update laws are dependent. The proper selection of the weight updated laws can guarantee that the function $V(k)$ is a Lyapunov function a unique global minimum. Thus the LABP is different from the gradient search-based algorithms which have fixed structure after the cost function is chosen. Their weights updated laws are only a mean to search for the global minimum and are independent of the cost function.

Remark 6.3: It can be seen, from the design procedure of the LABP in the section 6.3, the expressions (6.11)-(6.18), that only the input and output measurements are used for the design of the LABP in this paper. Thus the stochastic properties of the signals do not affect the performance the algorithm for TDNN. The main reason is that the optimization technique used in our paper is based on the Lyapunov stability theory not on the gradient search techniques. It is known that the gradient search–based optimization techniques are indeed affected by the stochastic properties of the signals. However, if the input disturbances are bounded random processes, the weight updated algorithm in TDNN can be directly designed using the input and output measurements based on the Lyapunov stability theory without considering the stochastic properties of the signals. This statement is akin to the design of Lyapunov stability based adaptive control systems and variable structure control systems in [20].

Remark 6.4: Stability and speed of convergence is very important in real-time applications, where time varying systems have to be tracked. Stability is also a significant problem when training dynamical neural networks. The stability of the weight updates BP algorithms cannot be guaranteed [28] and the instability can occur if the learning rate is too large. In LABP approach, the weight updated law in the LABP algorithm is designed based on Lyapunov stability theory. According to the Lyapunov theory [20], the stability of the error dynamics between the desired response and MLP output is guaranteed.

Remark 6.5: The weight updated laws $\Delta W_{J,I}^{(1,0)}(k)$, $\Delta W_{1J}^{L}(k)$ in (6.12), and (6.14), and $u(k)$ in (6.15) can be modified as follow to prevent the singularities due to zero values of $x_I(k)$ and $\Delta W_{1J}^{L}(k)$ and $S_J(k)$ becomes close to zero.

$$\Delta W_{1J}^{(2,1)}(k) = \frac{1}{S_J(k)+\lambda_1}\frac{1}{N(l=1)}\left[d(k) - \sum_{J=1}^{N(l=1)} W_{1J}^{(2,1)} S_J(k)\right] + \alpha e(k-1) \qquad (6.20)$$

$$\Delta W_{JI}^{(1,0)}(k) = \left[-W_{JI}^{(1,0)}(k-1) + \frac{1}{N(l=0)}\frac{1}{x_I(k)+\lambda_2} g_J(u(k))\right] \qquad (6.21)$$

where $$u(k) = \frac{1}{N(l=1)}\frac{1}{W_{1J}(k)+\lambda_3} d(k) \qquad (6.22)$$

The constants of λ_1, λ_2, λ_3 can be selected as follow: $0 < \lambda_1, \lambda_2, \lambda_3 < 1$. The smaller values of λ_1, λ_2 and λ_3 contribute smaller the error $e(k)$. Introduction of the constants λ_1, λ_2 and λ_3 may slightly limit the tracking properties of the proposed scheme when the LABP algorithm operates in the presence of extreme high frequency components or a sudden large disturbance in the system. These constants may limit the updated weights to become large enough to track the large changes in order to obtain an excellent performance. However, the extreme high frequency components or sudden large disturbances are not usually present and they are not desired in many applications. The most important issue is that the stability of the proposed scheme is still guaranteed with the introduction of the bounded disturbance. Simulation examples in the section 6.4 will further verified the effect of these constants.

Remark 6.6: One of other approaches is the second order type, including *Newton method* [1],[69], the *Broyden-Fletcher-Golgarb-Shanno*, the *Levenberg-Marquar*dt [73] and others. These methods converge with much less iterations than that required by the conventional BP. The extended Kalman learning algorithm developed by Singhal and Wu [72] and Pushkorius and Feldkamp [29], and the nonlinear recursive least-squares learning algorithm suggested by Kollias and Anastrassious [71] also roughly belong to this category. However, the computational and storage burden is increased quadratically with the number of weights because they have to calculate

and store the Hessian or Jacobian matrix. By observing the weights update law in the LABP (6.11)-(6.16), it is noted that the computation and storage requirements of LABP is much less than the aforementioned methods. The main reason is that there is no computation of Hessian matrix. The computational complexity of the LABP is also less than the algorithms such as genetic algorithms, learning automata and simulated annealing [33]-[35].

6.4 Lyapunov Stability-based Adaptive Backpropagation (LABP) for Recurrent MLP with TDL's Inputs-outputs

In the past few years many researches have focused their efforts on recurrent neural networks because of their attracting capability to exhibit dynamic behavior. They also represent a very powerful computational model, but designing proper architectures for a given problem and devising effective learning procedures is a very challenging task. Many researchers have recently emphasized their efforts on devising efficient algorithms, mainly based on optimization schemes such as gradient based algorithms, for learning the weights of recurrent MLP. Like for feedforward networks, these learning algorithms may get stuck in local minima during gradient descent, thus discovering sub-optimal solution. The proposed LABP in this paper might offer new approach for this problem encountered in the recurrent MLP. *Figure 6.3* shows the architecture of the recurrent MLP with TDL's input-output. The output neuron can be expressed as (6.23)

$$y(k) = \sum_{j=1}^{N(l=1)} W_{1j}^{(2,1)} S_j(k) \tag{6.23}$$

The output of the hidden layer neuron is

$$S_j(k) = F_j\left(\sum_{i=1}^{N(l=0)} W_{ji}^{(1,0)} \phi_i(k) \right) \tag{6.24}$$

where $j = 1, 2, \ldots, N(l=1)$ and $i = 1, 2, \ldots, N(l=0)$ and gives

$$y(k) = \sum_{j=1}^{N(L)} W_{1j}^{(2,1)} F_j\left(\sum_{i=1}^{N(l=0)} W_{ji}^{(1,0)} \phi_i(k)\right) \tag{6.25}$$

The inputs, $\phi_i(k)$ takes $\{x_i(k), x_i(k-1), \ldots, x_i(k-n), y_i(k-1), y_i(k-2), \ldots, y_i(k-m)$ with $i=1,2,\ldots N+M$, are passed to successively to the input layer of the MLP neural network.

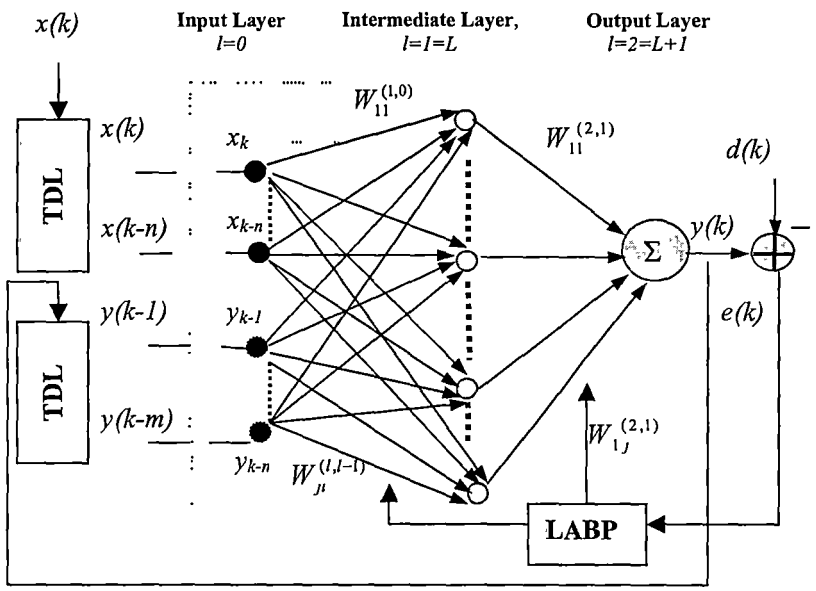


Figure 6.3: Recurrent MLP with TDL's Inputs and Outputs

6.4.1 LABP Recurrent Network Training Algorithm

The strategy for updating the network parameters is similar to LABP training algorithm in the section 6.2.2 and can be summarized as follow:

$$W_{1j}^{(2,1)}(k) = W_{1j}^{(2,1)}(k-1) + \Delta W_{1j}^{(2,1)}(k) \tag{6.26}$$

$$\Delta W_{1j}^{(2,1)}(k) = \frac{1}{S_j(k)} \frac{1}{N(l=1)} \left[d(k) - \sum_{j=1}^{N(l=1)} W_{1j}^{(2,1)} S_j(k) \right] + \alpha e(k-1) \tag{6.27}$$

and $$W_{ji}^{(1,0)}(k) = W_{ji}^{(1,0)}(k-1) + \Delta W_{ji}^{(1,0)}(k) \tag{6.28}$$

$$\Delta W_{ji}^{(1,0)}(k) = \left[-W_{ji}^{(1,0)}(k-1) + \frac{1}{N(l=0)} \frac{1}{\phi_i(k)} g_j(u(k)) \right] \tag{6.29}$$

where $$u(k) = \frac{1}{N(l=1)} \frac{1}{W_{1j}(k)} d(k) \tag{6.30}$$

$$g_j(\bullet) = F_j^{-1}(\bullet) \text{ and } 0 < \alpha < 1 \tag{6.31}$$

or $$\Delta W_{1j}^{(2,1)}(k) = \frac{1}{S_j(k) + \lambda_1} \frac{1}{N(l=1)} \left[d(k) - \sum_{j=1}^{N(l=1)} W_{1j}^{(2,1)} S_j(k) \right] + \alpha e(k-1) \tag{6.32}$$

$$\Delta W_{ji}^{(1,0)}(k) = \left[-W_{ji}^{(1,0)}(k-1) + \frac{1}{N(l=0)} \frac{1}{\phi_i(k) + \lambda_2} g_j(u(k)) \right] \tag{6.33}$$

where $$u(k) = \frac{1}{N(l=1)} \frac{1}{W_{1j}(k) + \lambda_3} d(k) \tag{6.34}$$

Remark 6.7: It is easy to see that the design of the LABP recurrent training algorithm is the same as the one given in theorem 6.1 if we replace the input x_i *(k)* with $\phi_i(k)$ in the section 6.3.

6.5 Simulation Examples

In this section, simulation examples that illustrate the performance of the LABP algorithm for adaptive filtering problem in the Chapter 2 are presented.

Example 1: LABP for TDNN or MLP with TDL's Inputs

The adaptive filter is implemented by a three-layer MLP. The input layer is composed of the TDL of the input signal, *x(k)*. The hidden layer is composed of 3 sigmoidal units. The output layer contained one linear unit that is connected to the hidden nodes. Both input-hidden and hidden-output layers have connection weights.

In the first case, no additive noise is considered in the simulation. *Figure 6.4a* has revealed the comparison of the output of MLP with LABP, $y(k)$ and the desired signal, $d(k)$. The square error, $e^2(k)$ is illustrated in *Figure 6.4b*. The weights of input-hidden and hidden-output layers are plotted in *Figure 6.4c*, *Figure 6.4d*, *Figure 6.4e* and *Figure 6.4f*. The resulted MSE (mean square error) is $1.69\text{x}10^{-4}$. Simulation has also shown the MSE is smaller if smaller values λ_1, λ_2, λ_3 are used.

For comparison, MLP trained by some first and second order gradient methods are performed. These methods are BP, BPM, Netwon, Guass-Netwon [70]. The results of BP and BPM are illustrated in *Figure 6.5a*, *Figure 6.5b*, *Figure 6.6a* and *Figure 6.6b*. The MSEs of BP and BPM are 0.0079 and 0.0073 respectively. Simulation results of Netwon, Guass-Netwon are not shown due to the ill-conditioning suffered by both methods during the computation. It is well-known that computing the Hessian matrix in Netwon method is computationally expensive and the Hessian matrix may not be positive definite at very point in the error surface. An alternative to Netwon method is Gauss-Netwon, but Gauss-Netwon may still have ill-conditioning if the matrix is close to or is singular. These problems have been experienced during the simulation computation in this particular example.

In the second example, the signal is corrupted by a uniformly distributed white noise sequence, $n(k)$ varying in the range [0,0.5] and gives SNR≈14 dB approximately. The corrupted signal is shown in *Figure 6.7a*. The comparison of the output of MLP with LABP, $y(k)$ and the desired signal, $d(k)$ are revealed in *Figure 6.7b*. *Figure 6.7c* illustrates the square error, $e^2(k)$. From the simulation results, the effect of additive noise is reduced greatly using the proposed LABP.

In the third example, the input signal has experienced a large input disturbance. *Figure 6.8* illustrates the corrupted input signal, $x(k)$. The bounded noise is analogous to previous example except a sudden large disturbance is introduced within iterations 1000-2000. *Figure 6.9a* and *Figure 6.9b* reveal the output signal, $y(k)$ and the square output error, $e^2(k)$ of the LABP. The weights of input-hidden and hidden-output layers when the input is subjected to the disturbance are plotted in *Figure 6.9c*, *Figure 6.9d*, *Figure 6.9e* and *Figure 6.9f*.

Simulations are also computed for the aforementioned methods for comparison, but we do not include these comparison results in the simulation section of the chapter because of the limit of the paper length. But we have addressed the good robustness of the LABP with respect to additive noise and large disturbance in the paper. In summary, the simulation results have revealed that the proposed LABP design has better performance in terms of error convergence, tracking ability and resistance of additive noise.

Example 2: LABP for Recurrent MLP with TDL's Inputs and Outputs

This example demonstrates the use of the LABP for the on-line recursive algorithm for adaptive filtering. The adaptive filter is implemented by a three layer MLP. The input layer is composed of the TDL of the input signal, $x_i(k)$ and feedback output signal, $y_i(k-1)$. The hidden layer is composed of 5 sigmoidal units. The original signal is corrupted by an additive noise, $n(k)$, that is a white uniformly distributed random variable with a range between 0 and 1. The adaptive neural filter is first simulated with LABP algorithm. *Figure 6.10a* reveals the comparison of the network output $y(k)$ and the desired response, $d(k)$. The effect of additive noise is reduced greatly and $y(k)$ follow the desired response $d(k)$ closely. MSE is indicated in *Figure 6.10b*. For the same setup, the adaptive neural filter is then trained with BPTT and its performance is shown in *Figure 6.11a*. Due to large signal to noise ratio, the additive noise cannot be eliminated well, thus a vivid distance between the filter output, $y(k)$ and the desired response, $d(k)$ is observed in *Figure 6.11a*. *Figure 6.11b* has revealed the MSE achieved by the BPTT.

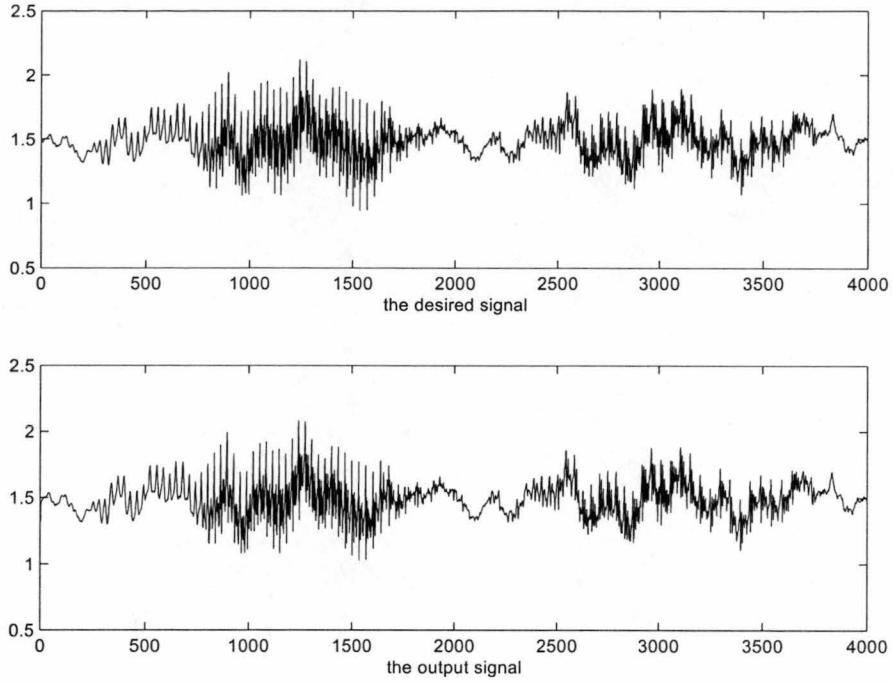


Figure 6.4a: LABP-The desired signal, *d(k)* & the MLP output, *y(k)* (no additive noise)

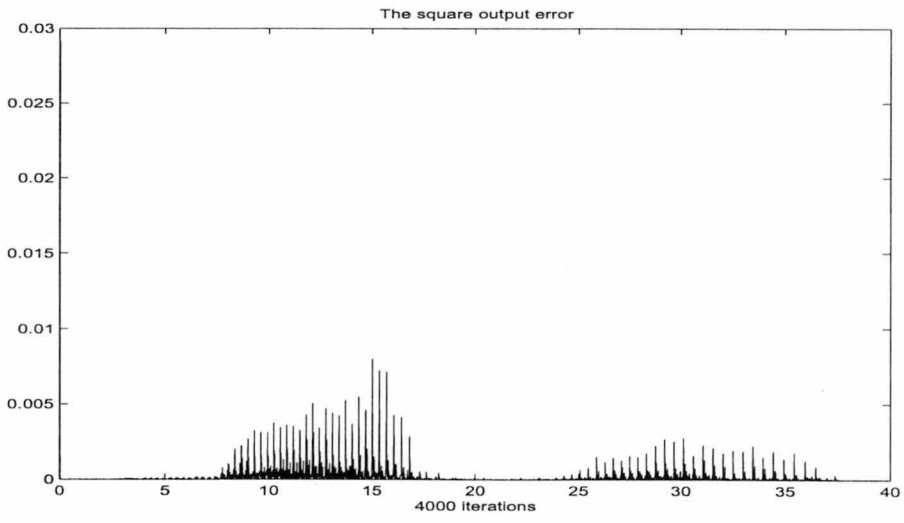


Figure 6.4b: LABP-The squared output error, $e^2(k)$ (no additive noise)

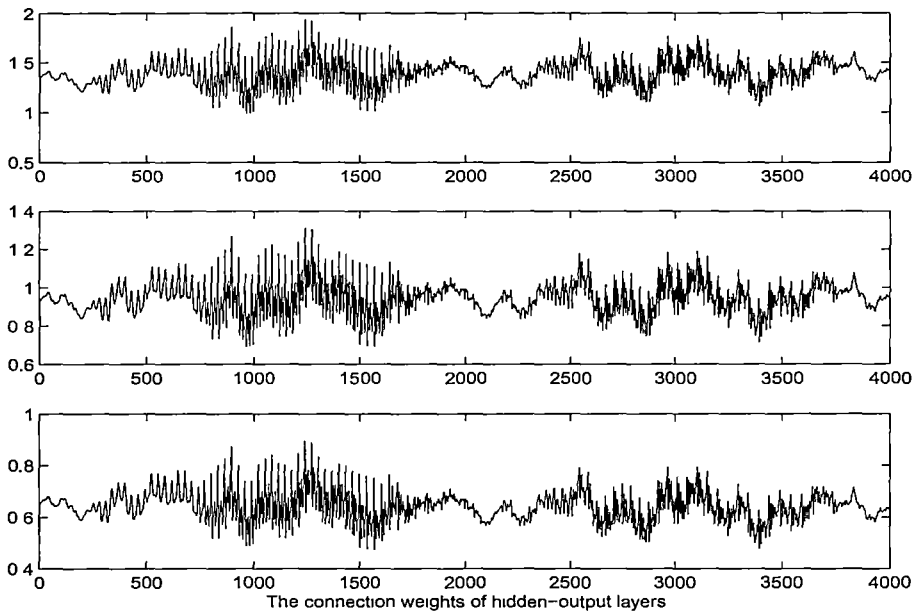


Figure 6.4c: LABP- The weights of hidden-output layers (no additive noise)

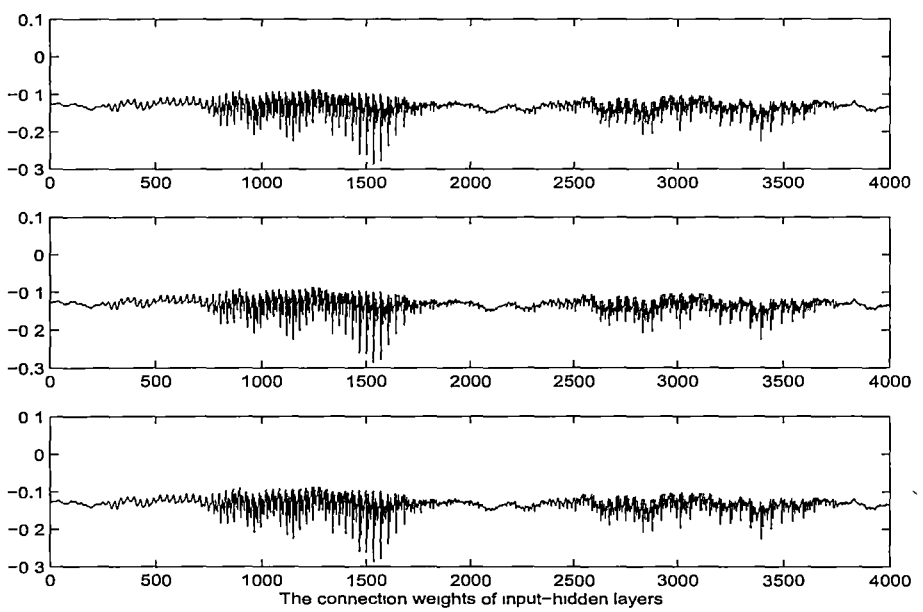


Figure 6.4d: LABP- The weights of input-hidden layers (1) (no additive noise)

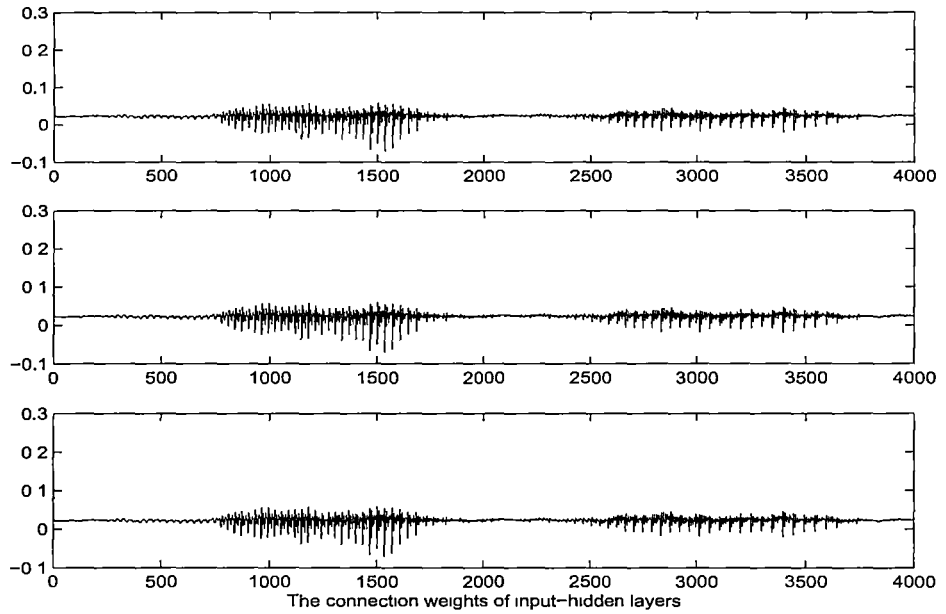


Figure 6.4e: LABP- The weights of input-hidden layers (2) (no additive noise)

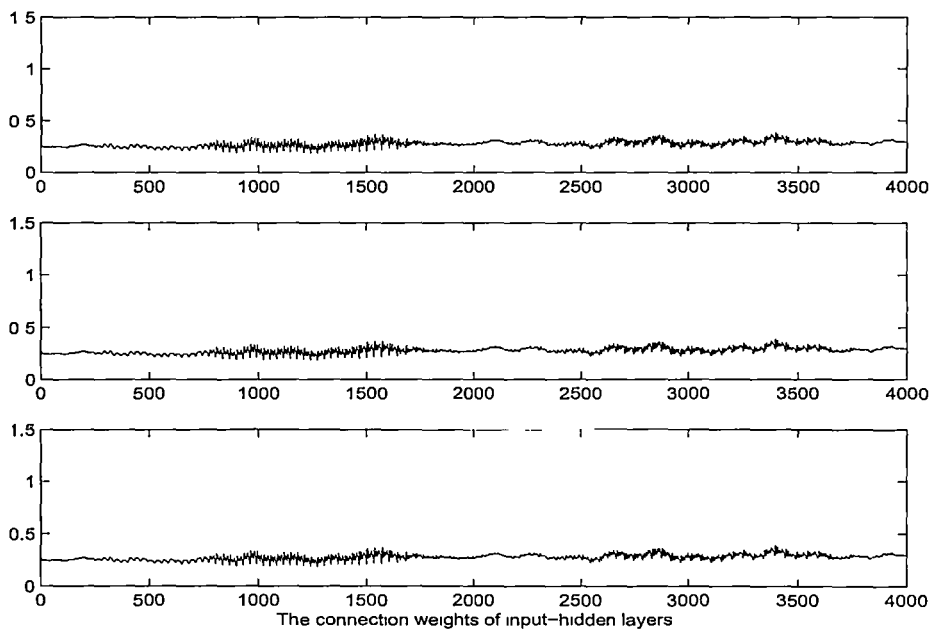


Figure 6.4f: LABP- The weights of input-hidden layers (3) (no additive noise)

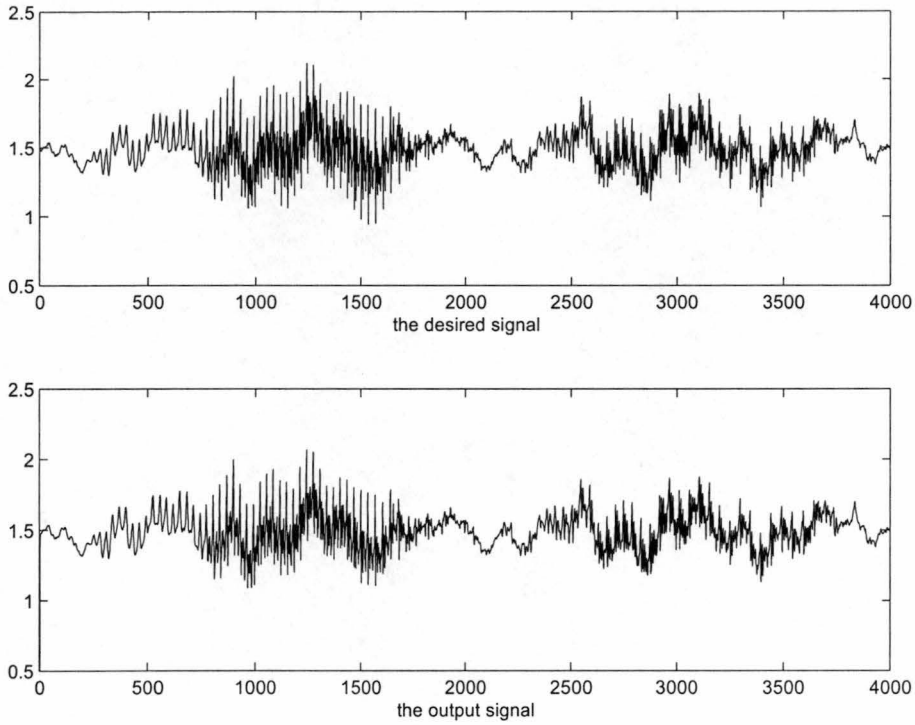


Figure 6.5a: BP-The desired signal, *d(k)* & the MLP output, *y(k)* (no additive noise)

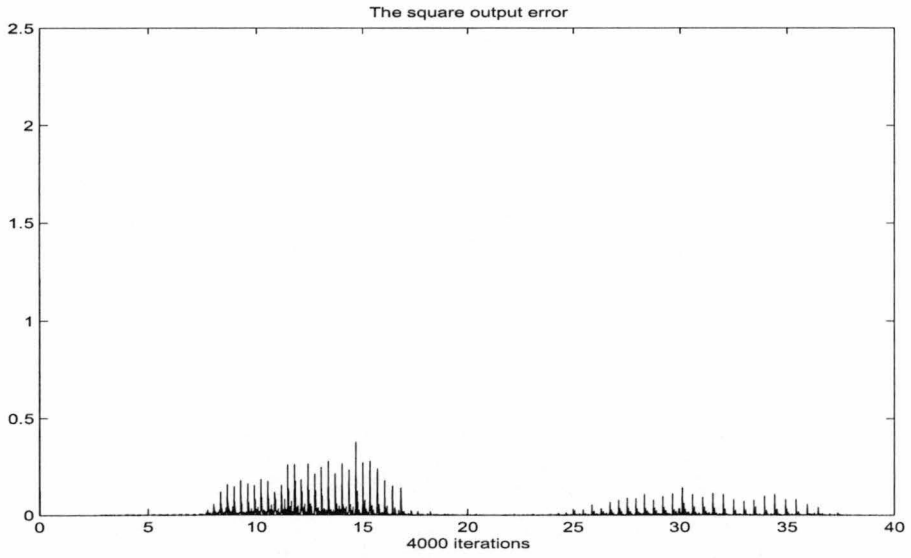


Figure 6.5b: BP-The squared output error, $e^2(k)$ (no additive noise)

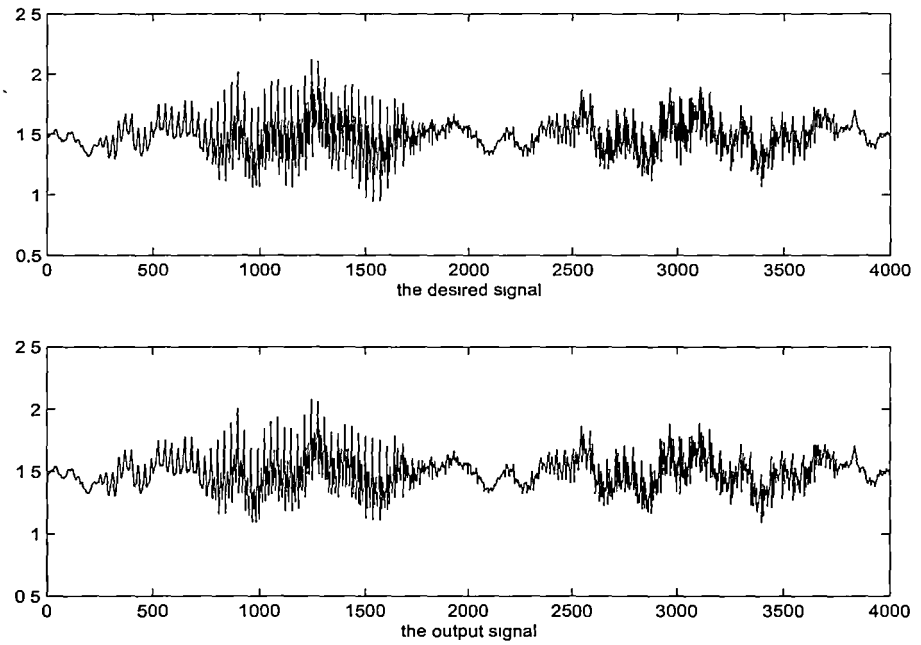


Figure 6.6a: BPM-The desired signal, *d(k)* & the MLP output, *y(k)* (no additive noise)

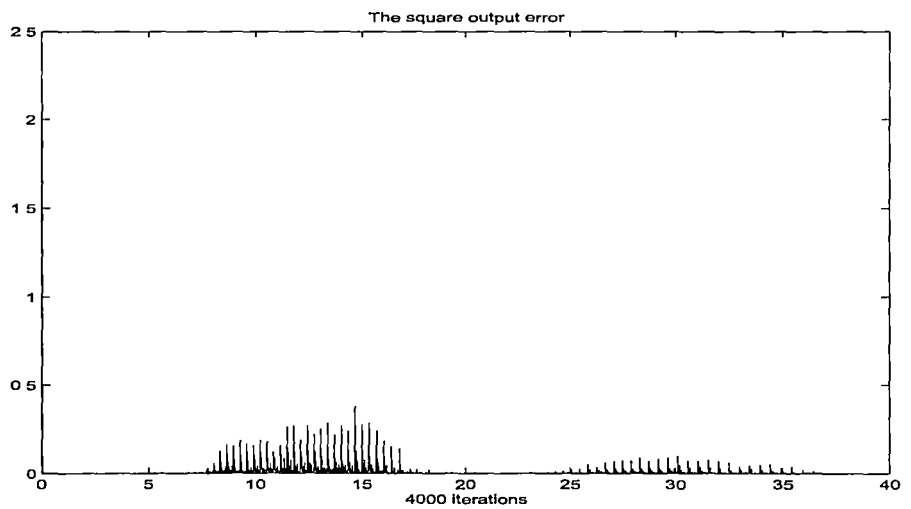


Figure 6.6b: BPM-The squared output error, $e^2(k)$ (no additive noise)

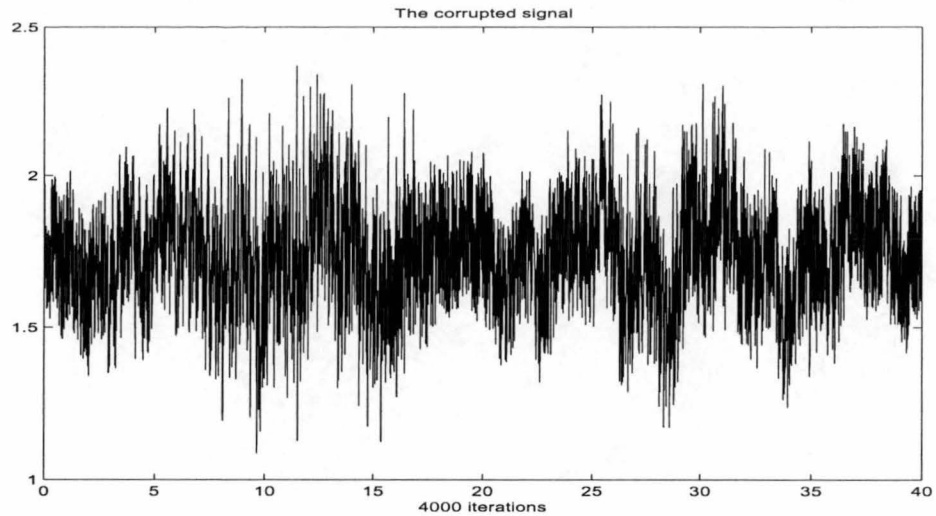


Figure 6.7a: The corrupted input signal, *x(k)* (additive noise)

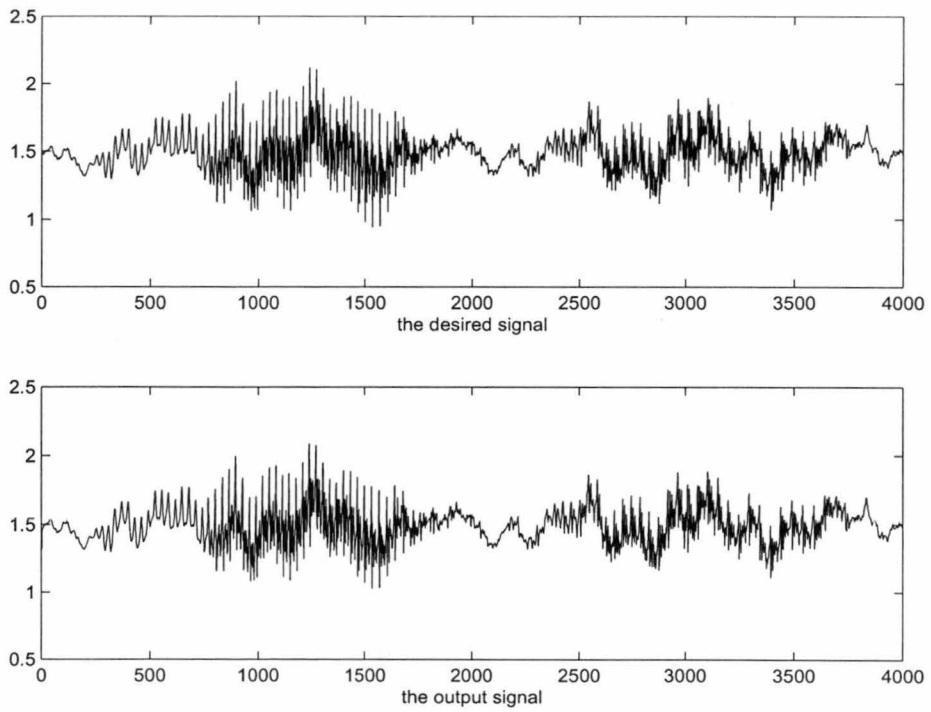


Figure 6.7b: LABP-The desired signal, *d(k)* & the MLP output, *y(k)* (additive noise)

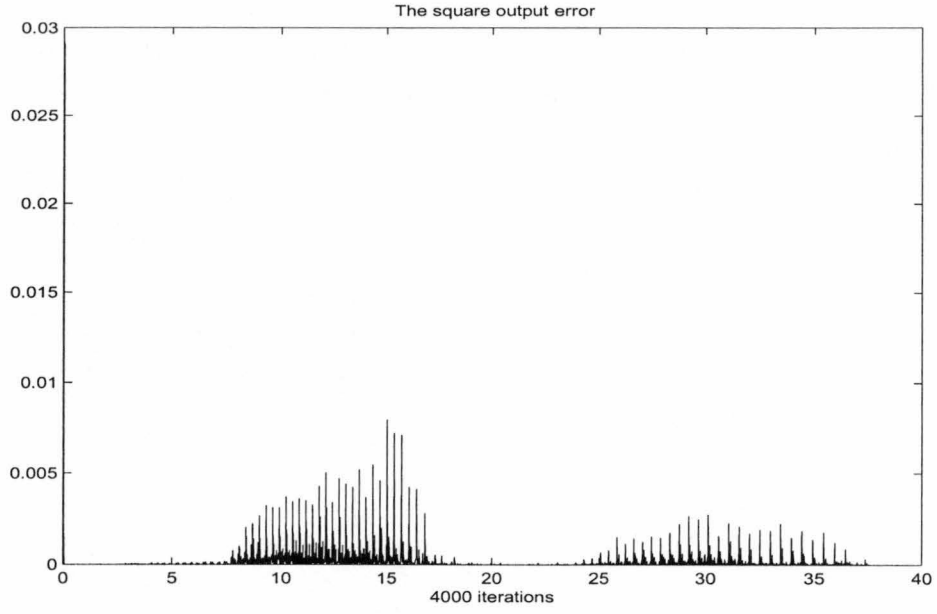


Figure 6.7c: LABP-The squared output error, $e^2(k)$ (additive noise)

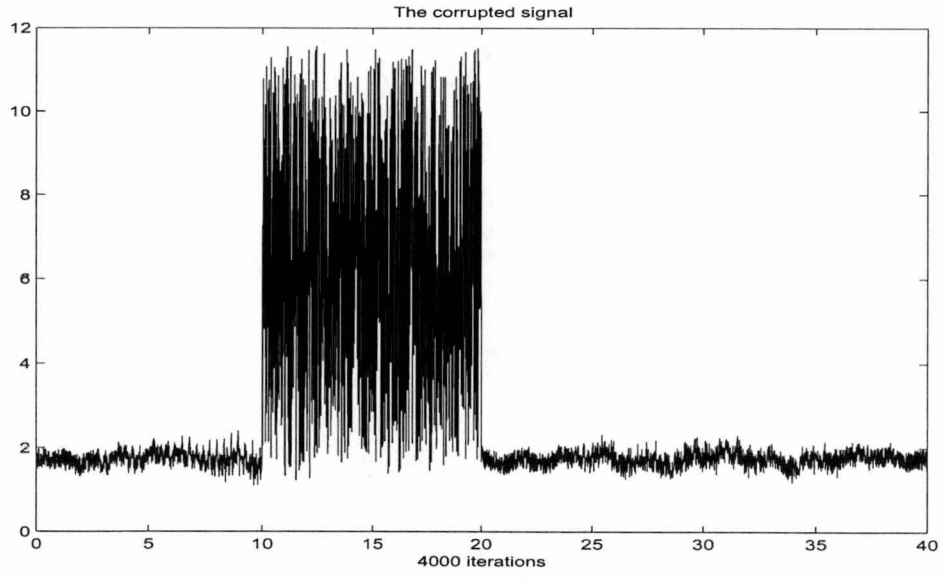


Figure 6.8: The corrupted input signal, $x(k)$ (large disturbance)

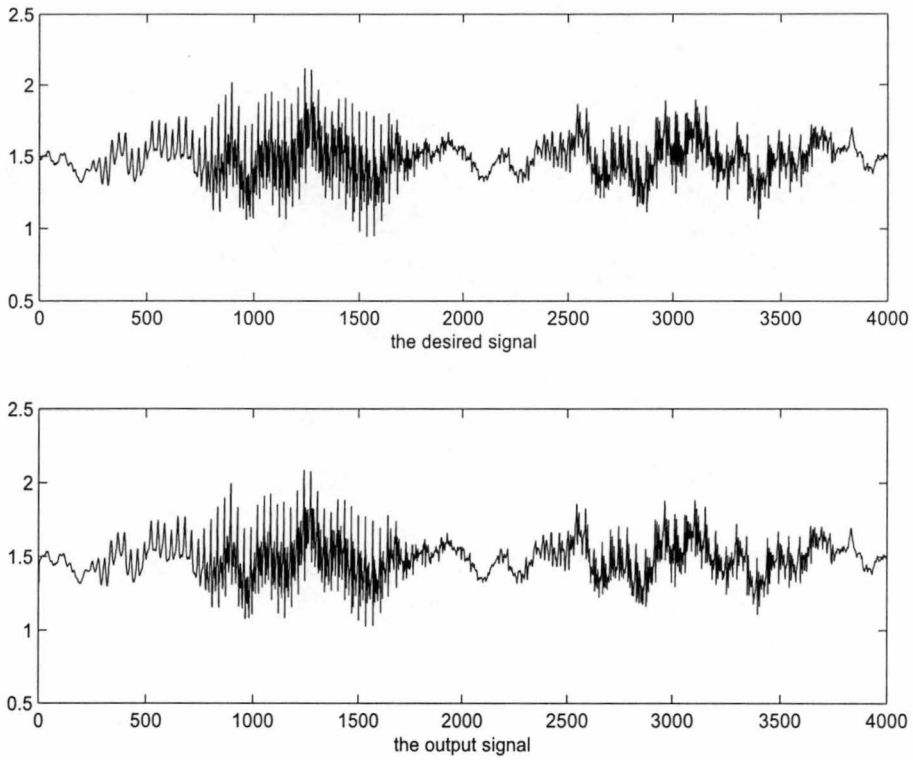


Figure 6.9a: LABP-The desired signal, *d(k)* & the MLP output, *y(k)* (large disturbance)

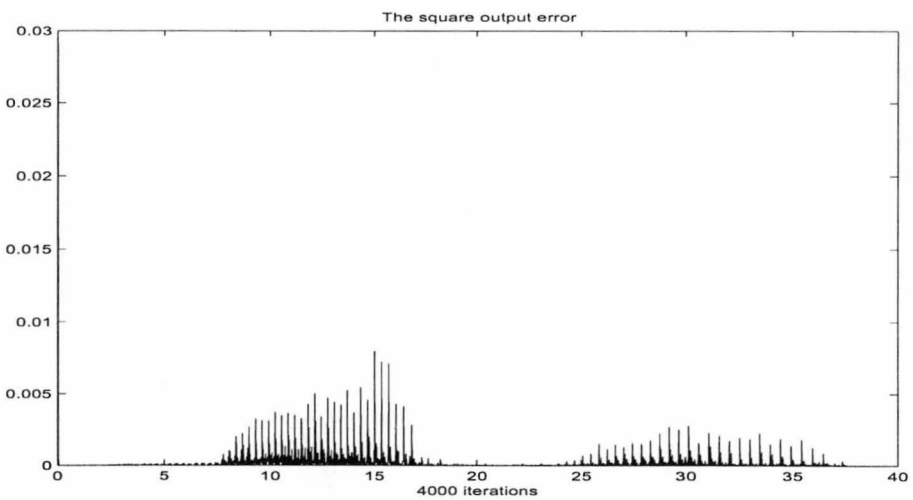


Figure 6.9b: LABP-The squared output error, $e^2(k)$ (large disturbance)

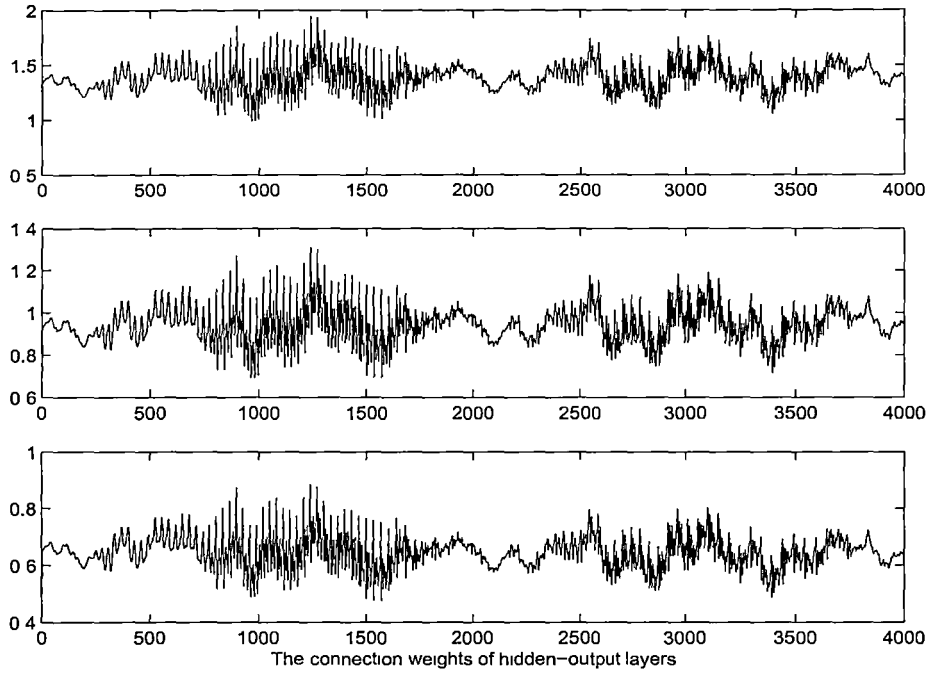


Figure 6.9c: LABP- The weights of hidden-output layers (large disturbance)

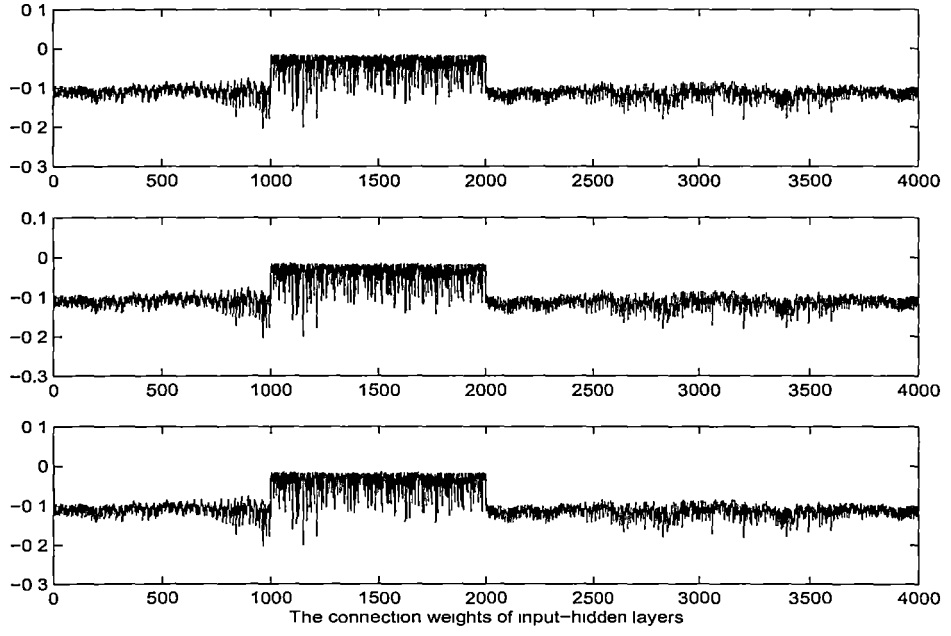


Figure 6.9d: LABP- The weights of input-hidden layers (1) (large disturbance)

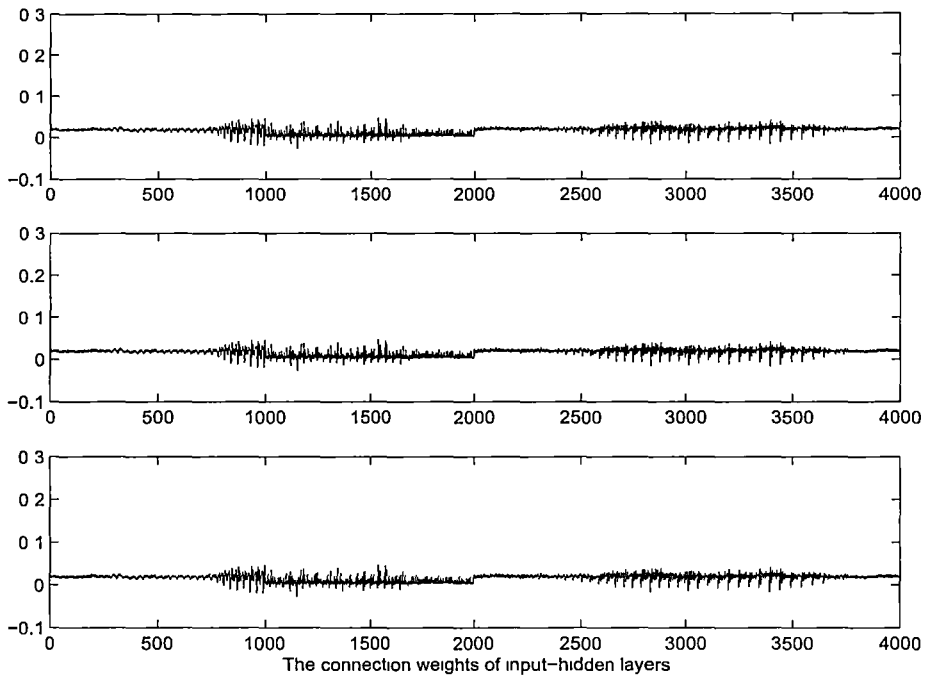


Figure 6.9e: LABP- The weights of input-hidden layers (2) (large disturbance)

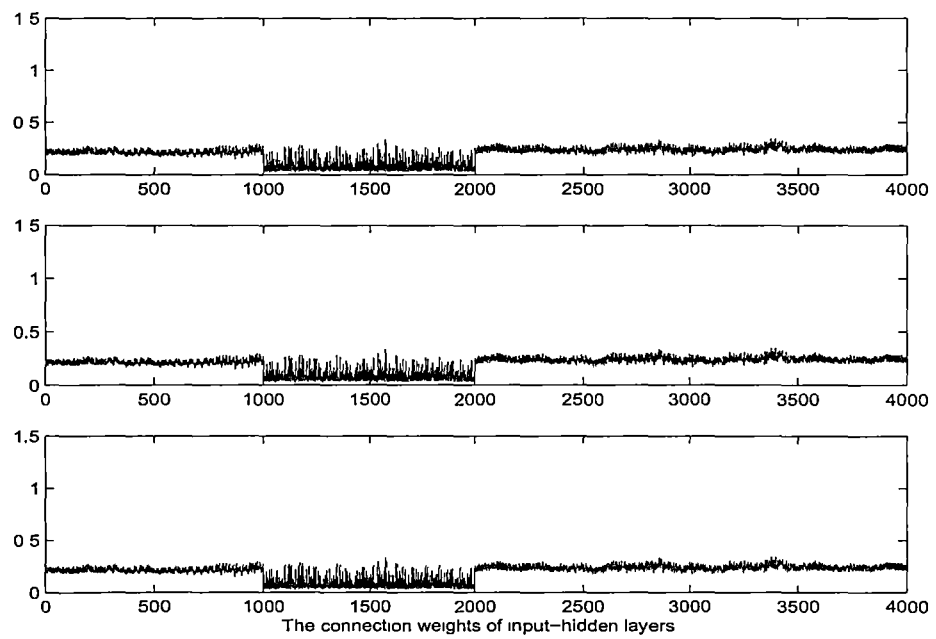


Figure 6.9f: LABP- The weights of input-hidden layers (3) (large disturbance)

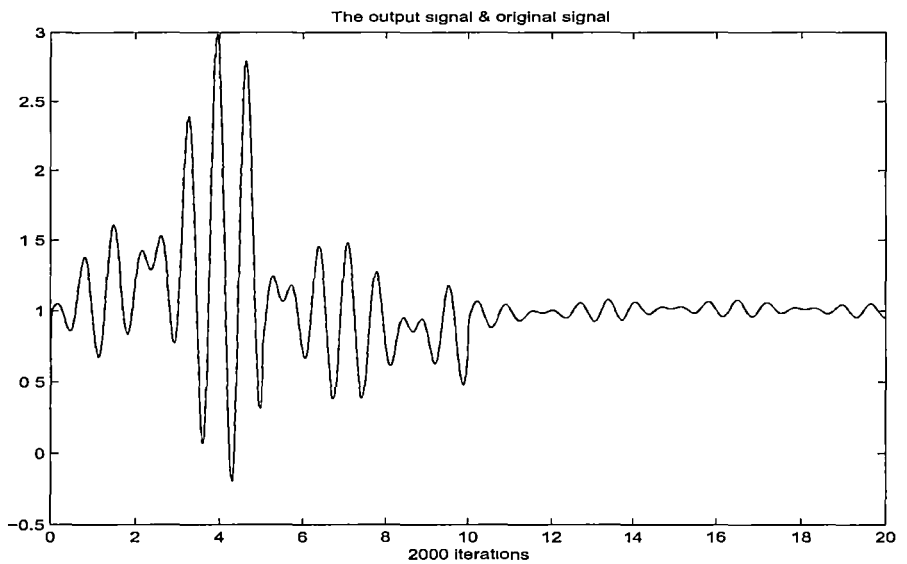


Figure 6.10a: LABP(recurrent)-The desired response, *d(k)* & filter output, *y(k)*

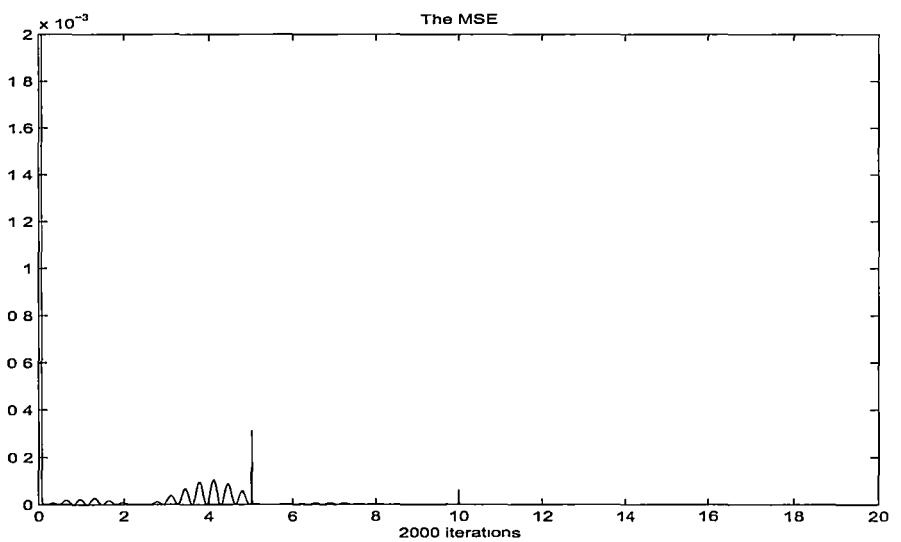


Figure 6.10b: LABP(recurrent)-The mean square error, MSE

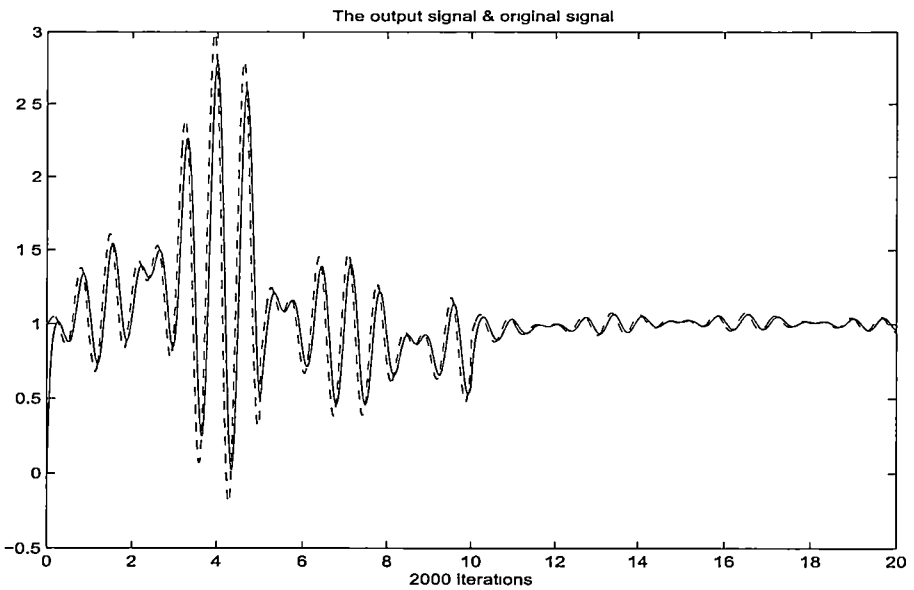


Figure 6.11a: BPTT-The desired response, *d(k)* & filter output, *y(k)*

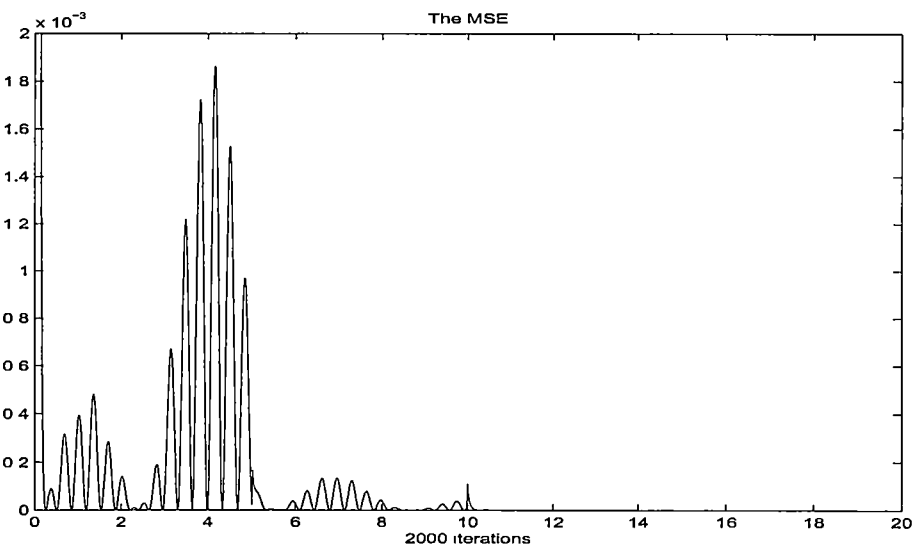
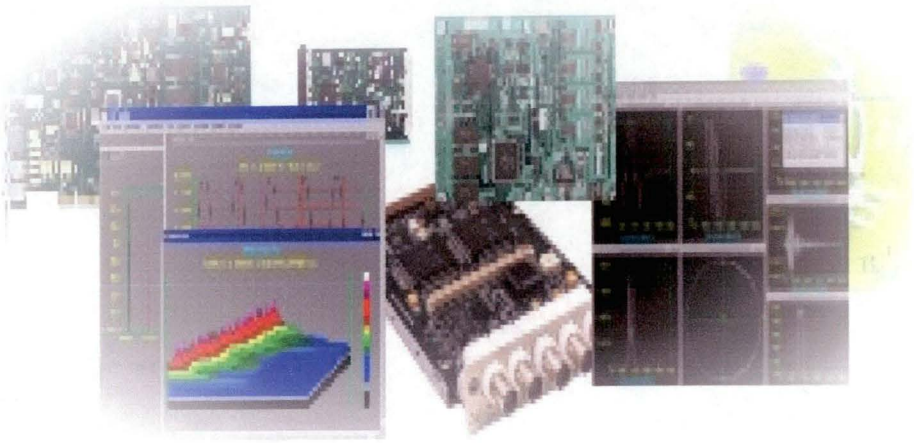


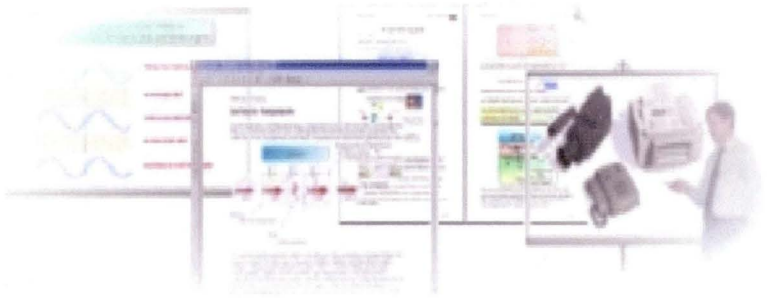
Figure 6.11b: BPTT-The mean square error, MSE

6.6 Conclusion

In conclusion, the theoretical and simulation results have suggested that Lyapunov stability based adaptive backpropagation (LABP) algorithm can be a new approach to the design of the neural network training algorithm. The proposed LABP algorithm is non-gradient based algorithm. Based on the Lyapunov theory, MLP's weights are adaptively adjusted so that the output error converges to zero asymptotically. This scheme is independent of stochastic properties of the signals. The derivation and design of the LABP is straightforward. The stability concern for the LABP algorithm is guaranteed by the Lyapunov Stability Theory. There are a number of open issues that exist with the proposed scheme. Theoretical analysis and further experimental work are required to further concrete the design of BP using Lyapunov theory. In particular, the following issues are seen as important: 1) theoretical and experimental works on the LABP for the MLP with more than one hidden layer. 2) Different Lyapunov functions and weight updated laws. The further research-based that can be carried out is to use different Lyapunov functions and different weight updated laws to further improve the convergence properties and the robustness properties of the LABP algorithm with respect to the bounded random disturbances. Therefore the LABP algorithm is an exciting and challenging area with a wide variety of applications, but much future work or research need to be done.

Chapter 7





Polynomial Signal Processing Using Lyapunov Theory

Chapter 7

Polynomial Signal Processing Using Lyapunov Theory

7.1 Introduction

The objective of this chapter is to present one area of nonlinear signal processing known as polynomial signal processing using Lyapunov theory. The first part of this chapter presents a fast, less computation complexity and stable adaptive polynomial filters. We only focus on the following polynomial models: (1) *Volterra model* that the nonlinear system output signal can be related to the input signal through a truncated Volterra series expansion. (2) *Bilinear model* that involves recursive nonlinear difference equation. The second part of the chapter considers another realization of nonlinear Volterra filter using *parallel-cascade* structure. Parallel-cascade realizations implement higher order Volterra systems as a parallel connection of multiplicative combinations of lower order truncated Volterra systems.

All the proposed techniques in this chapter have excellent convergence and their stability are guaranteed by the Lyapunov stability theory. These schemes are independent of signals' stochastic properties. They have less or comparable computational complexity compared to some conventional polynomial filters. Simulation examples have demonstrated the performance of these new designs.

Chapter 7 is structured as follows. In section 7.2, the Lyapunov adaptive Volterra filter (LAVF) is proposed. This is then followed by the Lyapunov adaptive Bilinear filter in section 7.3. The theoretical derivation is further supported by the simulation

examples in the section 7.4. In section 7.5, a new computation efficient adaptive algorithm for parallel-cascade truncated Volterra system is presented. Lastly, the concluding remark is presented in section 7.6.

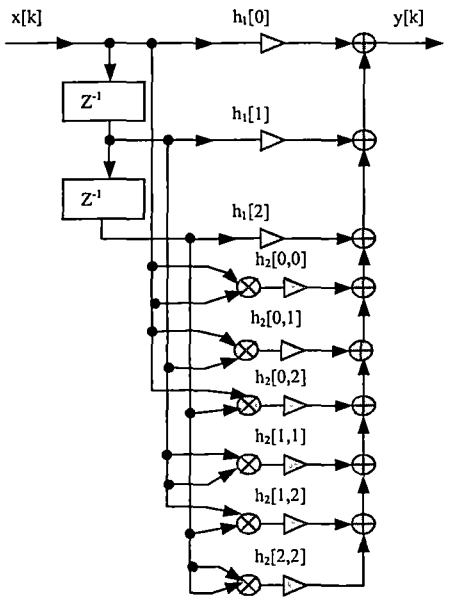


Figure 7.1: 2nd truncated Volterra system

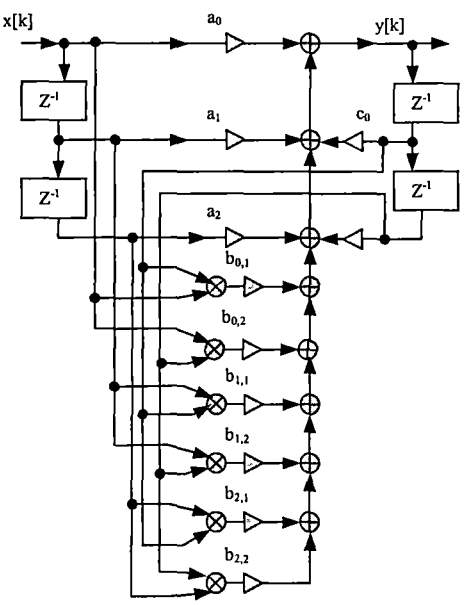


Figure 7.2: Bilinear system, $N= 3$

7.2 Lyapunov Adaptive Volterra Filters (LAVF)

In the Chapter 3, we have proposed *Lyapunov theory-based adaptive filtering* (LAF) techniques for FIR and IIR filters. Now we consider the realization of LAF using polynomial structure. In this section, we merely focus on the system representations using a second order Volterra series expansion. In the Volterra series representation of systems, which is an extension of linear system theory, the output *y(k)* of any casual, discrete-time, time-invariant nonlinear system can be represented as a function of the input sequence *x(k)*. Considering an SISO system, the Volterra series expansion is given by

$$y(k) = h_0 + \sum_{m_1=0}^{\infty} h_1(m_1)x(k-m_1) + \sum_{m_1=0}^{\infty}\sum_{m_2=0}^{\infty} h_2(m_1,m_2)x(k-m_1)x(k-m_2) + \ldots$$

$$+ \sum_{m_1=0}^{\infty}\sum_{m_2=0}^{\infty}\ldots\sum_{m_p=0}^{\infty} h_p(m_1,m_2\ldots,m_p)x(k-m_1)x(k-m_2)\ldots x(k-m_p) + \ldots\ldots \qquad (7.1)$$

where $h_p(m_1, m_2,\ldots m_p)$ is known as the *p-th* order Volterra kernel of the system. In filtering application, the infinite series expansion in (7.1) is not useful, hence the truncated Volterra series expansions of the form (7.2) is employed.

$$y(k) = \sum_{m_1=0}^{N-1} h_1(m_1)x(k-m_1) + \sum_{m_2=0}^{N-1}\sum_{m_1=0}^{N-1} h_2(m_1,m_2)x(k-m_1)x(k-m_2) + \cdot \\ + \sum_{m_p=0}^{N-1}\cdots\sum_{m_1=0}^{N-1} h_p(m_1,m_2\ ,m_p)x(k-m_1)x(k-m_2)\ldots x(k-m_p) \tag{7.2}$$

Note that there are $O(N^P)$ coefficients in this polynomial expansion (ie. the number of coefficients is proportional to N^P). It is well known the major drawback for the Volterra system model in (7.2) is that the complexity of implementing filters using this model can be very large even for moderately large values of N and P. Consequently, most of the practical application of systems employing Volterra series expansions involve low-order models. *Figure 7.1* shows the block diagram of an adaptive Volterra filter. For simplicity, a second order Volterra series expansion is considered. The adaptive filter tries to estimate the desired response signal, $d(k)$ using a second-order truncated Volterra series expansion in the input signal $x(k)$.

$$y(k) = \sum_{m_1=0}^{N-1} h_1(m_1)x(k-m_1) + \sum_{m_2=0}^{N-1}\sum_{m_1=0}^{N-1} h_2(m_1,m_2)x(k-m_1)x(k-m_2) \tag{7.3}$$

For notational simplicity as well as ease of performance analysis, it is usual to write the algorithm using vector notation, thus (7.3) can be rewritten as

$$y(k) = H^T(k)X(k) \tag{7.4}$$

where $H(k) = [h_1(0,k), h_1(1,k),\ldots, h_1(N-1;k), h_2(0,0;k), h_2(0,1;k),\ldots,$

$h_2(0,N-1;k), h_2(1,1;k), \ldots h_2(N-1,N-1;k)]^T$

$X(k) = [x(k), x(k-1), \ldots, x(k-N+1), x^2(k), x(k)\,x(k-1),\ldots,$

$x(k)\,x(k-N+1), x^2(k-1),\ldots x^2(k-N+1)]^T$

Using the results in Chapter 3, we have the following updated law for the LAVF adaptive filter:

$$H(k) = H(k-1) + g(k)\alpha(k) \tag{7.5}$$

$$\alpha(k) = d(k) - H^T(k-1)X(k) \tag{7.6}$$

$$g(k) = \frac{X(k)}{\| X(k) \|^2}\left(1 - \kappa\frac{|e(k-1)|}{|\alpha(k)|}\right) \tag{7.7}$$

or $$g(k) = \frac{X(k)}{\lambda_1 + \| X(k) \|^2}\left(1 - \kappa\frac{|e(k-1)|}{\lambda_2 + |\alpha(k)|}\right) \tag{7.8}$$

where $0 \leq \kappa < 1$.

Remark 7.1: It is easy to notice that the stability analysis of the error dynamics, convergence analysis of the LAVF adaptive filter are the same as those given in *Theorem 3.1-3.3* in Chapter 3.

Remark 7.2: The nonlinear Volterra filtering with LMS [74] suffers from slow convergence due to large eigenvalues spread. The other approach RLS converges fast but exhibits unstable behavior and suffers from ill-conditioning.

Remark 7.3: Due to the fact the number of kernel increases exponentially as the filter order increases and this leads computation complexity also increases exponentially, a less computation adaptation algorithm is needed. The computation complexity of LAVF updated algorithm is less than that of RLS $O(N^4)$ for 2^{nd} order Volterra filtering [74].

7.3 Lyapunov Adaptive Bilinear Filters (LABF)

The major problem associated with Volterra series representation of nonlinear systems is that a large number of coefficients are required to characterize many nonlinear processes. Consequently it is important to search for alternative representations that may be more parsimonious in their use of coefficients. One such model is that in which the input-output relationship is governed by a recursive nonlinear difference equation of the type

$$y(k)=\sum_{i=1}^{M}P_i(y(k-1),y(k-2),\ ,y(k-N+1),x(k),x(k-1),\ .,x(k-N+1)) \tag{7.9}$$

The simplest of the nonlinear systems in this category is the *bilinear system* :

$$y(k)=\sum_{i=1}^{N-1}c_i y(k-i)+\sum_{i=0}^{N-1}\sum_{j=1}^{N-1}b_{i,j}y(k-j)x(k-i)+\sum_{i=0}^{N-1}a_i x(k-i) \tag{7.10}$$

In spite of the simplicity, this is an important nonlinear model since it can be shown under relatively mild conditions that a large class of nonlinear systems including Volterra systems can be approximated with arbitrary precision using bilinear system models with finite number of coefficients [74]. *Figure 7.2* shows the block diagram of a bilinear filter for the case when *N=3*.

Just as linear IIR filters can model many linear systems with more parsimony than FIR filters, there are a large number of nonlinear systems that can be approximated by nonlinear feedback models using a relatively small number of parameters. In such situations, once can expect that the adaptive bilinear filters can be implemented with good computational efficiency. Another attractive feature of the bilinear system models is that they can be used to approximate any Volterra system with arbitrary precision under fairly general conditions [75]. Due to these advantages, bilinear system models have found various applications, including those in control system, signal processing, biological systems, etc.

An overview of continuous time bilinear system models and their applications can be found in [76]. In spite of the potential benefits of such system models, very little work has been donc on adaptive filters employing nonlinear feedback models. Among the very few published works are [78]-[79]. The results in [78]-[79] involve direct-form structures and employ the conventional recursive least square adaptation algorithm or its variations, which are computationally very complex. Fast versions of such algorithms will almost certainly suffer from numerical problems. Reference [79] contains a Kalman filter type algorithm for adaptive bilinear filtering when the only unknown parameters are the noise statistics. The approach in [80] performs a Gram-Schmidt orthogonalization of the data. However implementing using this method for the structure shown in (7.10) requires high computation. Paper [78]

discusses an algorithm involving the simpler LMS adaptive filter. Again, such algorithms are known for their slow and input-dependent convergence rates. Lattice structures are attractive because of the existance of fast and numerically stable adaptive algorithms.

In this section, a stable and fast bilinear adaptive filters is presented. The proposed algorithm in section 7.2 can be easily applied to bilinear filter. The expression (7.10) with input and coefficient vectors, $H^*(k)$, $X^*(k)$can be rewritten as

$$y(k) = H^{*T}(k)X^*(k) \tag{7.11}$$

where $H^*(k) = [c_1(k), c_2(k),\ldots, c_{N-1}(k), b_{0,1}(k),\ldots, b_{N-1,N-1}(k), a_0(k),\ldots,a_{N-1}(k)]^T$

$X^*(k) = [y(k-1), y(k-2), \ldots, y(k-N+1), x(k)y(k-1), \ldots,$

$x(k-N+1)y(k-N+1), x(k),\ldots, x(k-N+1)\ ]^T$

Then we have the following updated law for the LAVF adaptive filter:

$$H^*(k) = H^*(k-1) + g(k)\alpha(k) \tag{7.12}$$

$$\alpha(k) = d(k) - H^{*T}(k-1)X^*(k) \tag{7.13}$$

$$g(k) = \frac{X^*(k)}{\| X^*(k) \|^2}\left(1 - \kappa\frac{|e(k-1)|}{|\alpha(k)|}\right) \tag{7.14}$$

$$g(k) = \frac{X^*(k)}{\lambda_1 + \| X^*(k) \|^2}\left(1 - \kappa\frac{|e(k-1)|}{\lambda_2 + |\alpha(k)|}\right) \tag{7.15}$$

where λ_1, λ_2 are small positive numbers and $0 \leq \kappa < 1$.

Remark 7.4:Due to the fact that the feedback is included, the stability concerns for the adaptive algorithms which are gradient based is no longer guaranteed. In these LAVF and LABP filters, the stability is guaranteed by Lyapunov stability theory.

7.4 Simulation Examples for LAVF and LABF Filters

In this section, we present the results of several experiments that demonstrate the good properties of the LAVF and LABF as well as verify the theoretical analysis presented earlier. The first part of the simulation example demonstrates the performance of the LAVF and LABF when an additive noise is introduced at the filter input. Simulation of the same setup with RLS, LMS is also accomplished for comparison. The second part of the simulations illustrates the performance of LAVF and LABF to the nonlinear system identification.

Example 1: Adaptive Filtering

Adaptive filtering with Lyapunov adaptive Volterra filter (LAVF) - In this example, *Figure 7.3a* illustrates a speech signal is corrupted with the additive noise *n(k)*, uniform distributed random noise {0 1}. The adaptive gain is updated according to the expression (7.8) and $\lambda_1 = \lambda_2 = 0.01$, $\kappa = 0.001$. The result illustrated in *Figure 7.3b* shows the comparison of the reference signal *d(k)* and the filter output signal *y(k)*. It is seen that the output of the LAVF can follow the desired reference signal closely and the effects of noise is well eliminated. *Figure 7.3c* reveals the square output error, $e^2(k)$. For a comparison study, simulations of same setup for the Volterra filters with RLS, LMS algorithms are also presented. The results in *Figure 7.4a* (forgetting or weighting factor, $\rho = 0.2$) and *Figure 7.5a* reveals the output signal of RLS and LMS respectively. They have higher noise level compared to that of LAVF by observing the square output error, $e^2(k)$ in *Figure 7.4b* and *Figure 7.5b*. The Volterra filter with LMS tends to have weak performance in the high noise situation. Hence LAVF has fast convergence speed, good tracking property and is highly stable.

Adaptive filtering with Lyapunov adaptive Bilinear filter (LABF) – The corrupted input signal, *x(k)* and bounded noise are analogous to previous example. *Figure 7.6a, 7.6b* reveal the output signal, *y(k)* and the square output error, $e^2(k)$ of LABF respectively. These results have shown good performance of LABF filter.

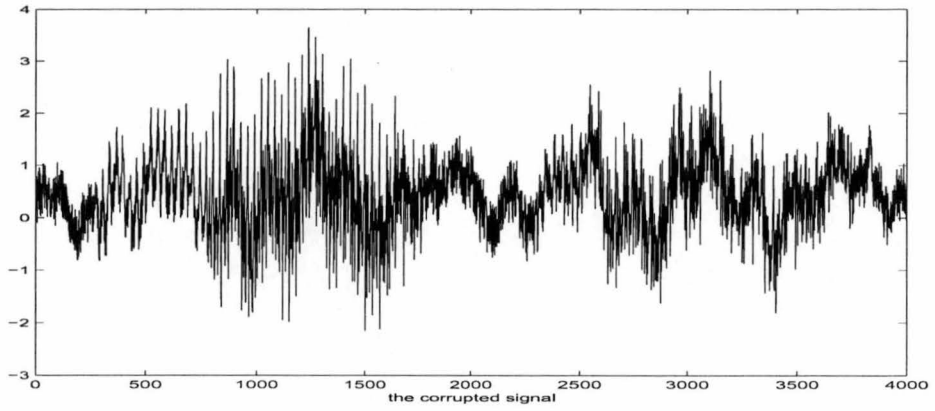


Figure 7.3a: The corrupted input signal

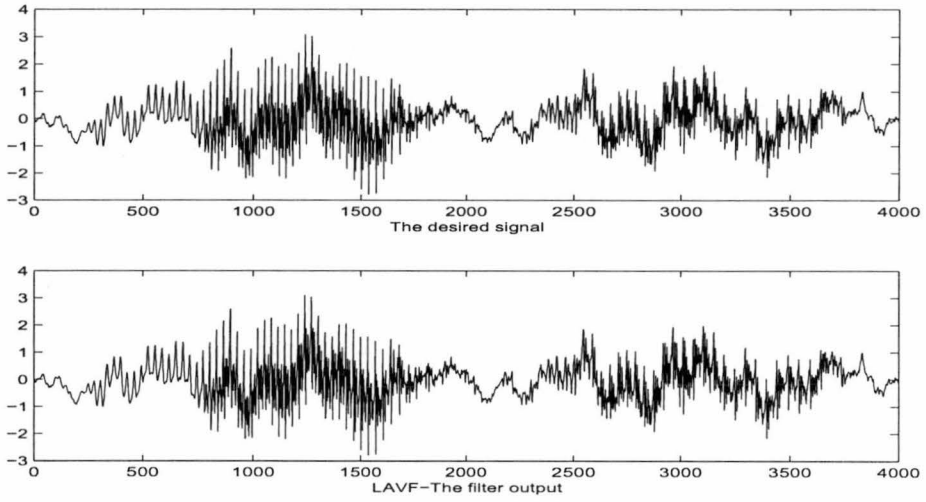


Figure 7.3b: LAVF-Desired output ,*d(k)* & filter output ,*y(k)*

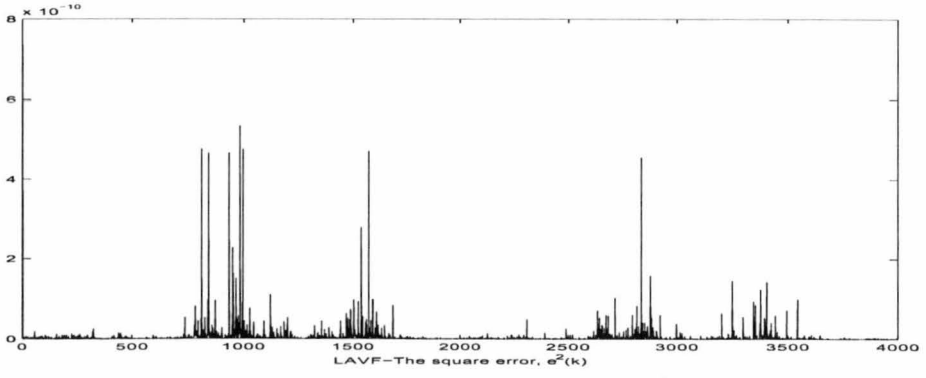


Figure 7.3c: LAVF-The square error, $e^2(k)$

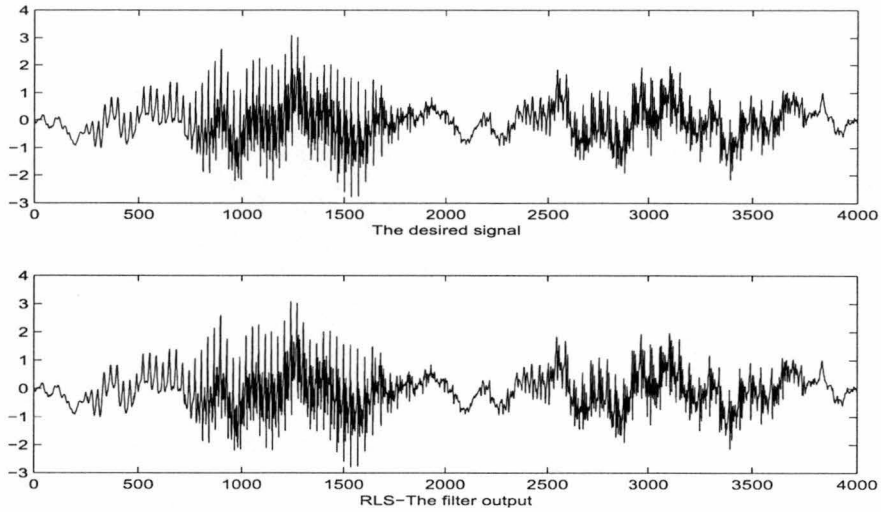


Figure 7.4a: RLS-Desired output, *d(k)* & filter output, *y(k)*

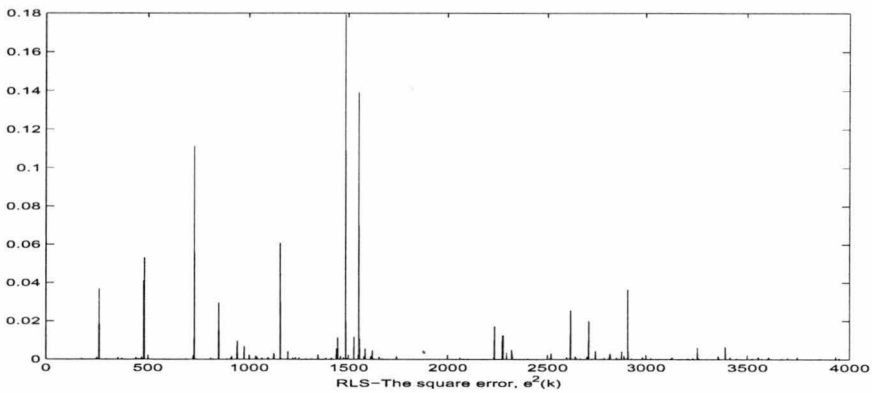


Figure 7.4b: RLS-The square error, $e^2(k)$

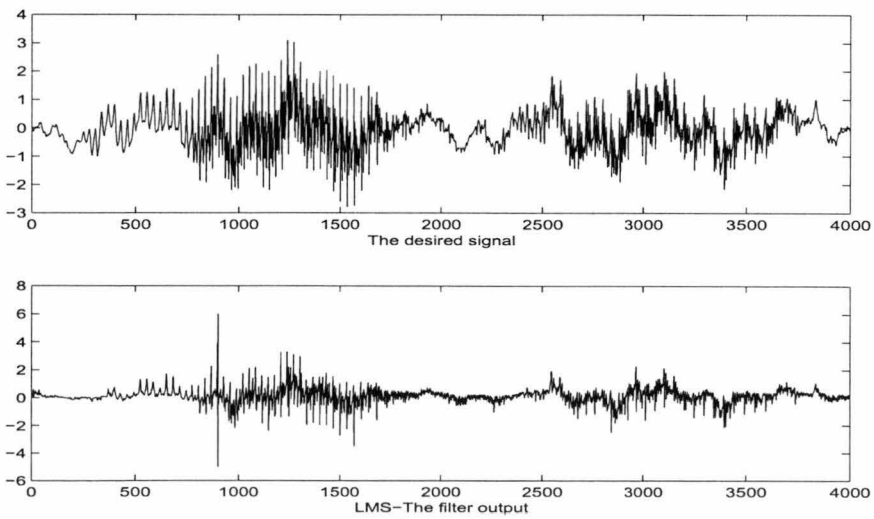


Figure 7.5a: LMS-Desired output ,*d(k)* & filter output ,*y(k)*

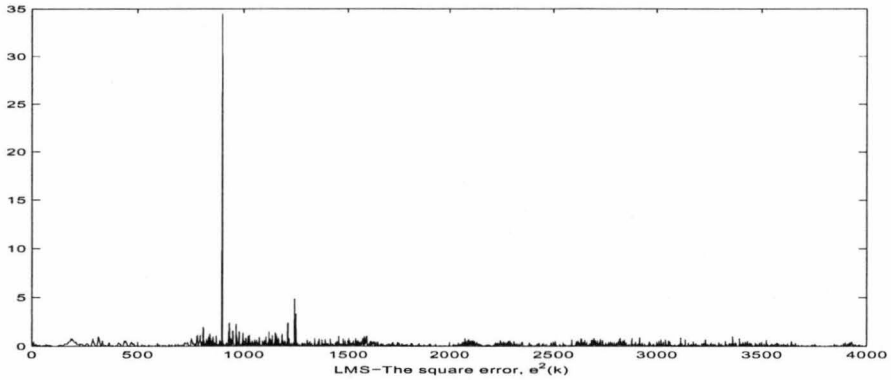


Figure 7.5b: LMS-The square error, $e^2(k)$

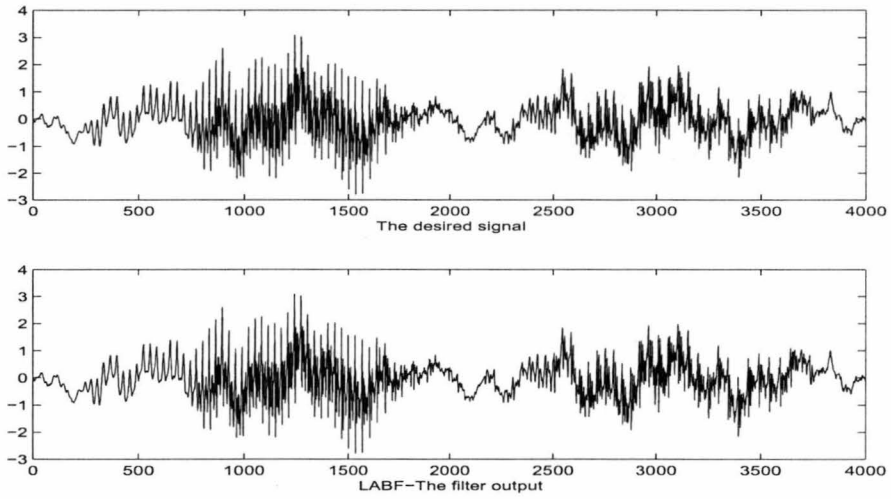


Figure 7.6a: LABF-Desired output, $d(k)$ & filter output, $y(k)$

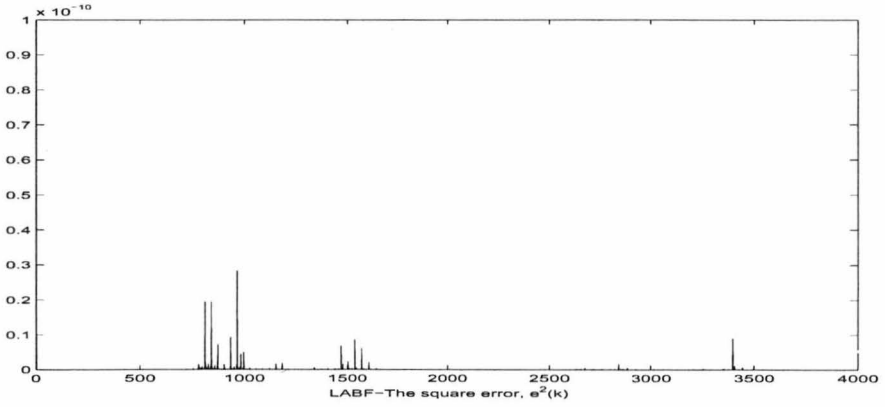


Figure 7.6b: LABF-The square error, $e^2(k)$

Example 2: Nonlinear System Identification

This example evaluates the performance of the proposed scheme when the underlying system model is different from the second-order Volterra system model used in the development of the adaptive filter. The problem considered here is that of identifying a nonlinear channel using the adaptive second-order Volterra filter illustrated in *Figure 7.7a*. The nonlinear channel is a simplified model of a digital transmission represents one of the most important cases of a digital communication employing a nonlinear channel [80]. The memoryless nonlinear device is an AM/AM converter whose characteristics are shown in *Figure 7.7b*.

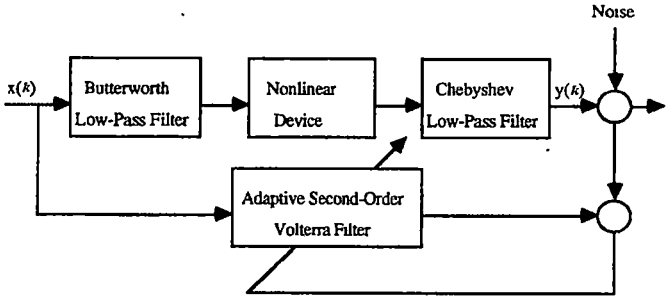


Figure 7.7a: Adaptive Filter To Identify A Nonlinear Transmission System

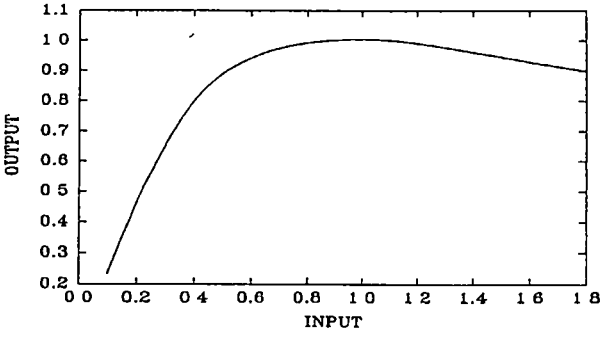


Figure 7.7b: Input-Output Characteristics of The AM/AM Converter

The transfer functions of the fourth-order low pass Butterworth and Chebyshev filters, denoted as $H_B(z)$ and $H_C(z)$ are given by

$$H_B(z)=\frac{(0.078+0.1559z^{-1}+0.078z^{-2})(0.0619+0.1238z^{-1}+0.0619z^{-2})}{(1-1.3209z^{-1}+0.6327z^{-2})(1-1.0486z^{-1}+0.2961z^{-2})} \tag{7.16}$$

and

$$H_C(z)=\frac{(0.4638-0.4942z^{-1}+0.4638z^{-2})(0.183+0.1024z^{-1}+0.183z^{-2})}{(1-1.2556z^{-1}+0.6891z^{-2})(1-0.7204z^{-1}+0.1888z^{-2})} \tag{7.17}$$

respectively. Both filters have a cutoff frequency 0.1 cycles/sample. The input signal *x(k)* is uniformly distributed on the interval [0.12 1.78] so that the AM/AM converter is operated at saturation region most of the time. *Figure 7.8a* shows the output of the LAVF and the output of the nonlinear system over the 5000 samples. The square output error is displayed in *Figure 7.8b*. Compared to the simulation results in [80] that the steady-state MSE's ≈ 0.00131-0.10623 by time averaging the ensemble averages in the range [9000 10000], LAVF has smaller square output error ≈ 10^{-10}. The further smaller square output error can obtained by using the smaller λ_1, λ_2, κ. It appears that the proposed algorithm works well in this situation even though the structure of the adaptive filter is completely different from that of the system model.

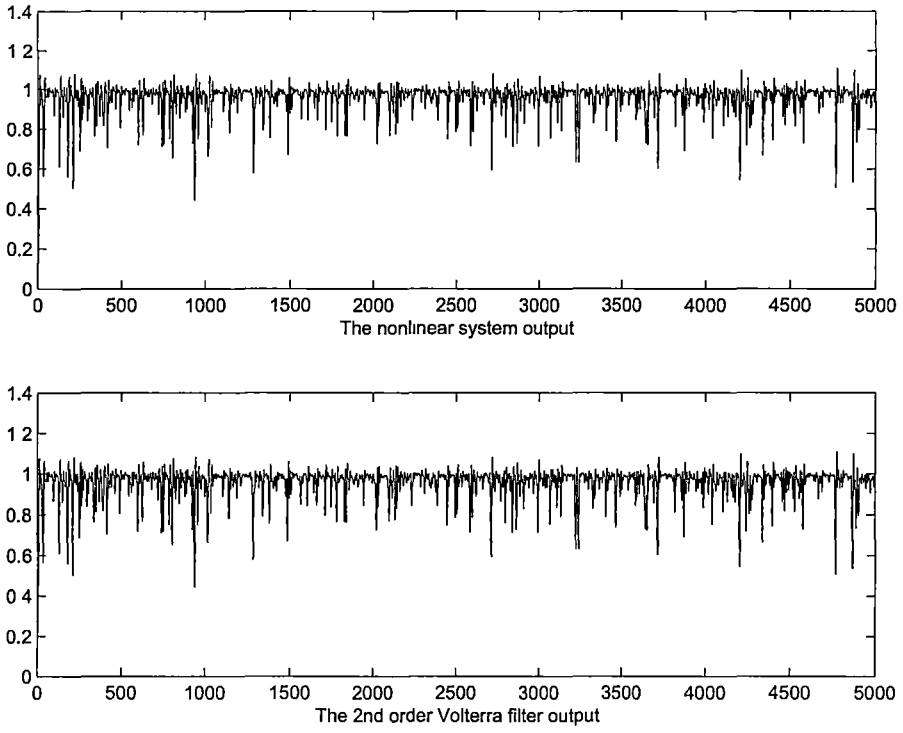


Figure 7.8a: The nonlinear transmission system output and the LAVF output

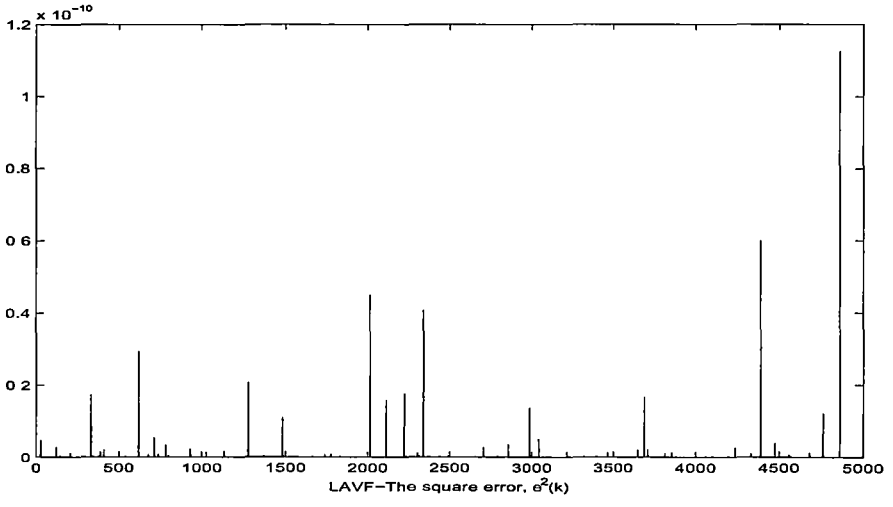


Figure 7.8b: LAVF-The square output error, $e^2(k)$

7.5 A New Computation Efficient Adaptive Algorithm For Parallel-Cascade Truncated Volterra System

Another the realization of nonlinear Volterra filter is *parallel-cascade* realization illustrated in *Figure 7.9*. Parallel-cascade realizations implement higher order Volterra systems as a parallel connection of multiplicative combinations of lower order truncated Volterra systems. Each branch in the *Figure 7.9* consists of lower order Volterra systems combined in a multiplicative manner. This structure and its algorithms are attractive because of the modularity of the parallel-cascade realizations to approximate nonlinear systems efficiently using a reduced number of branches. This realization and its variations have several advantages over direct-form realizations. It is conceptually and computationally simpler to implement lower order Volterra system components than higher order Volterra systems. The realization resulted in modular interconnections of low-order Volterra systems enabling efficient implementation of higher order Volterra systems using VLSI circuits. The second advantage relates to efficiency of implementation obtained through approximations. The parallel-cascade realizations provide a systematic method of approximating nonlinear systems by discarding less relevant branches in the realization [81],[82].

Several researchers have employed parallel-cascade filter [81]-[84]. Adaptive parallel-cascade filters for quadratic systems models were presented in [83] and [84]. The structure of [83] is not constrained to result in a unique solution to the estimation problem and the filter exhibits relatively slow convergence behavior. The work in [84] constrains the filter structure to provide convergence to a unique solution but it requires appropriate training to select its initial settings. Recently, authors in [81],[82] have presented the parallel-cascade realizations and approximations of truncated Volterra systems and applied this concept to high order Volterra filter. They claimed that this filter is capable of converging to a unique solution and does not require the use of a training signal to initialize the algorithm. They have designed the LMS and NLMS adaptive parallel-cascade Volterra algorithms. In the LMS parallel-cascade structure, singular value, LU or LDL^T decompositions are employed. This results in three different sets of weights update expressions for the time dependence components. However, the speed of convergence and the steady-

state characteristics are controlled by the positive constant or step-size. Because of the nonlinearities in the system model, derivation of the stability bounds for the step-size is a very difficult problem. The step-size selection is a simpler task for the NLMS parallel-cascade algorithm. However it is assumed that the level of measurement noise in the desired response signal and the level of nonstationary in the operating environment are relatively low. Furthermore authors have argued that the estimation error is bounded for certain choice of the step-size as long as the input signal and the desired response signal are bounded in some sense. There is no theoretical analysis on the step-size bound.

7.5.1 Parallel-cascade Realization of Truncated Volterra Systems

The output of a homogenenous and casual pth order Volterra system with N -sample memory is related to its input as

$$y(k) = \sum_{m_2=0}^{N-1} \sum_{m_2=m_1}^{N-1} \cdots \sum_{m_p=m_{p-1}}^{N-1} h_p(m_1, m_2 ..., m_p) \times x(k-m_1)x(k-m_2)...x(k-m_p) \qquad (7.18)$$

where $h_p(m_1,m_2, ..., m_p)$ represents the pth order Volterra kernal of the system.

A pth-order Volterra system can be realized using l-th order and $(p-l)$-th order Volterra systems as illustrated in *Figure 7.1*. The input-output relationship in the expression (7.18) can be compactly written using matrix notation as

$$y(k) = X_{N,l}^T(k) H_{N,l,p-l}(k) X_{N,p-l}(k) \qquad (7.19)$$

In the abovc cxpression, the column vector $X_{N,p}(n)$ has $\binom{N+p-1}{p}$ elements and contains all possible pth-order product signals of the form $x(k-m_1)x(k-m_2) \ldots x(k-m_p)$, where $0 \le m_1 \le m_2 \le \ldots \le m_p \le N-1$. Let $x(k-m_1)x(k-m_2) \ldots x(k-m_l)$ be the element of $X_{N,l}(k)$, and let $x(k-n_1)x(k-n_2) \ldots x(k-n_{p-l})$, where $0 \le n_1 \le n_2 \le \ldots \le n_{p-l} \le N-1$, be the jth element of $X_{N,p-l}(k)$. Then, $H_{N,l,p-l}$ is a coefficient matrix of dimension of $\binom{N+l-1}{l} \times \binom{N+(p-l)-1}{p-l}$ such the (i,j)-th element scales $x(k-m_1)x(k-m_2) \ldots x(k-m_l)\ x(k-n_1)x(k-n_2) \ldots x(k-n_{p-l})$ in the expression (7.18).

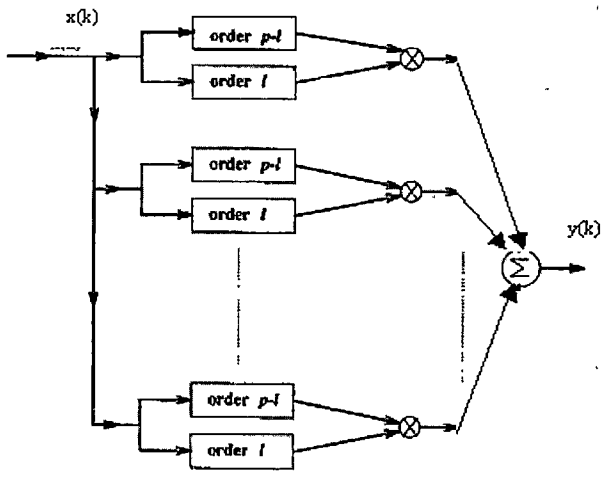


Figure 7.9: Parallel-cascade realization of a *p*th-order truncated Volterra system

7.5.2 The Design of The Lyapunov Theory-based Adaptive Parallel-Cascade Volterra Filter

The design of the adaptive filter using Lyapunov theory is described by the following theorem 7.1.

Theorem 7.1: For the given desired response $d(k)$, if the weight vector $H_{N,l,p-l}(k)$ of the filter $y(k) = X_{N,l}^T(k) H_{N,l,p-l}(k) X_{N,p-l}(k)$ is updated as follows

$$H_{N,l,p-l}(k) = H_{N,l,p-l}(k-1) + g(k)\alpha(k)$$

and
$$g(k) = \frac{X_{N,l}(k) X_{N,p-l}^T(k)}{\| X_{N,l}(k) \|^2 \| X_{N,p-l}(k) \|^2} \left(1 - \kappa \frac{|e(k-1)|}{|\alpha(k)|} \right) \tag{7.20}$$

where $0 \le \kappa < 1$, then the error $e(k) = d(k) - y(k)$ asymptotically converges to zero.

Proof: Define a Lyapunov function

$$V(k) = e^2(k) \tag{7.21}$$

$$\Delta V(k) = V(k) - V(k-1)$$

$$
\begin{aligned}
&= e^2(k) - e^2(k-1) \\
&= \left(d(k) - X_{N,l}^T(k) H_{N,l,p-l}(k) X_{N,p-l}(k)\right)^2 - e^2(k-1) \\
&= (d(k) - X_{N,l}^T(k)\left(H_{N,l,p-l}{}^T(k-1) + g(k)\alpha(k)\right) X_{N,p-l}(k))^2 - e^2(k-1) \\
&= (d(k) - X_{N,l}^T(k) H_{N,l,p-l}(k-1) X_{N,p-l}(k) - X_{N,l}^T(k) g(k)\alpha(k) X(k))^2 - e^2(k-1) \\
&= (\alpha(k) - X_{N,l}^T(k) g(k) \alpha(k) X_{N,p-l}(k))^2 - e^2(k-1) \\
&= \alpha^2(k)\left(1 - X_{N,l}^T(k) g(k) X_{N,p-l}(k)\right)^2 - e^2(k-1)
\end{aligned} \tag{7.22}
$$

Using the expression (7.20) in the expression (7.22), we have

$$\Delta V(k) = -(1-\kappa^2)e^2(k-1) < 0 \tag{7.23}$$

Remark 7.5 The deficiency of (7.20) is that the values of $X_{N,l}{}^T(k)$, $X_{N,p-l}(k)$ or $\alpha(k)$ may be zero and rise singularities problem is also noticed. Therefore the adaptation gain may be modified as the expression (7.24) to avoid singularities.

$$g(k) = \frac{X_{N,l}(k) X_{N,p-l}^T(k)}{\| X_{N,l}(k) \|^2 \| X_{N,p-l}(k) \|^2 + \lambda_1} \left(1 - \kappa \frac{| e(k-1) |}{| \alpha(k) | + \lambda_2}\right) \tag{7.24}$$

where λ_1, λ_2 are small positive numbers, $0 \le \kappa < 1$, then the error $e(k)$ asymptotically converges.

Remark 7.6: Due to the fact the number of kernel increases exponentially as the filter order increases and this leads computation complexity also increases exponentially. The proposed scheme computational complexity is less than that of the existing parallel-cascade truncated Volterra filters proposed in [81],[82].

Remark 7.7: Only the preliminary simulation results of the Lyapunov theory-based adaptive parallel-cascade Volterra filter, $p=2$, $l=1$, for nonlinear adaptive filtering are presented. The parallel-cascade realization of a pth-order truncated Volterra system for adaptive filtering will be further researched in the future. The simulation detail will not included here. A bounded additive noise: $0 < n(k) < 0.2$ is introduced at the system input. *Figure 7.10* illustrates the desired signal, $d(k)$ and the parallel-cascade Volterra system output, $y(k)$ is shown in *Figure 7.11*.

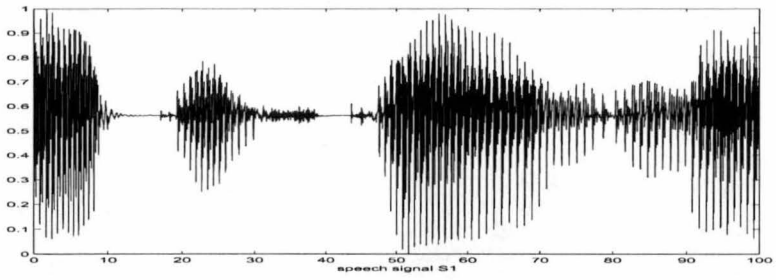


Figure 7.10: The desired signal or response, *d(k)*

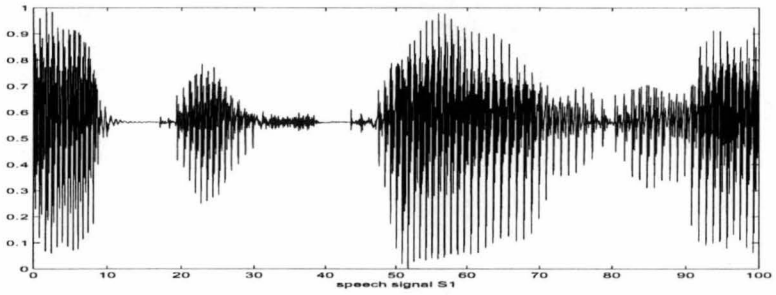
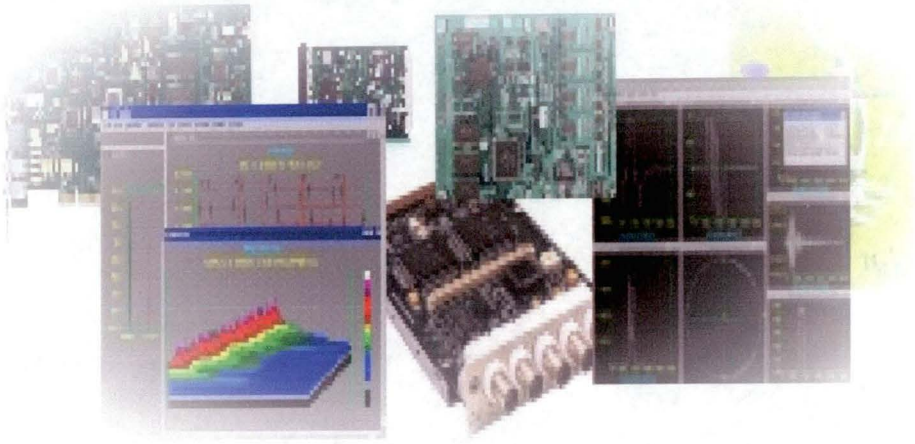


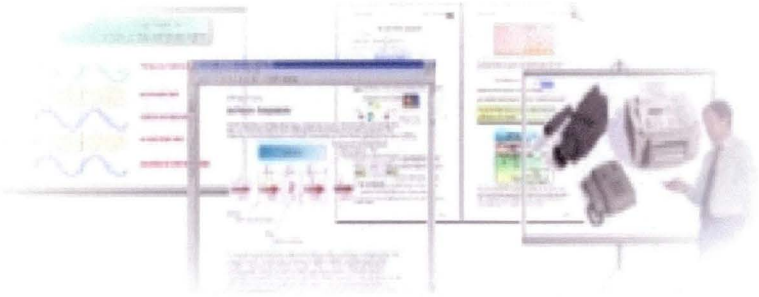
Figure 7.11 The Lyapunov theory-based adaptive parallel-cascade Volterra filter output, *y(k)*

7.6 Conclusion

Two efficient, less computation complexity, good tracking and stable adaptive nonlinear polynomial filters: Volttera and Bilinear filters designed based on Lyapunov theory are presented. The emphasis in the first part of the chapter is adaptive filter based on system models using truncated Volterra series expansions. This is followed by the bilinear adaptive filtering using recursive nonlinear system models. The theoretical analysis and simulations have indicated that the LAVF and LABF have fast convergence speed, highly stable compared to RLS and LMS with Volterra or bilinear model. The second part of the chapter has presented algorithm for adaptive truncated Volterra filters employing parallel-cascade structures. Parallel-cascade realizations implement higher order Volterra systems as a parallel connection of multiplicative combinations of lower order truncated Volterra systems. Only the theoretical analysis is performed for this scheme. There are several other issues that require further study. One is the implementation issue, such as those involving exploitation of parallelisms and modularities in the structure of the adaptive filter have not addressed in this chapter.

Chapter 8





Other Adaptive Filtering Techniques Using Lyapunov Theory And Applications

Chapter 8

Other Adaptive Filtering Techniques Using Lyapunov Theory And Applications

8.1 Introduction

This chapter introduces other different techniques besides those proposed in Chapter 3-Chapter 6. These techniques include (1) A new concurrent algorithm for adaptive filtering in parallel signal processing. (2) Complex-valued Lyapunov theory-based adaptive filtering. (3) A new approach in feedforward active noise control using Lyapunov stability theory. (4) A hybrid nonlinear neural predictor and its application to nonlinear and noisy time series prediction. Most of these methods are the modification of the scheme presented in Chapter 3-6 to suit particular applications.

This chapter is organized as follows. Section 8.2 presents a *concurrent Lyapunov theory-based adaptive filtering* (CLAF). In section 8.3, a *complex concurrent Lyapunov theory-based adaptive filtering* (Complex-LAF) is proposed. Section 8.4 suggests two algorithms called *Filtered-X Lyapunov theory-based* algorithm (FXLYP), *Filtered-U Lyapunov theory-based* algorithm (FULYP), and a overall on-line modeling techniques using *Lyapunov theory-based adaptive filtering* (LAF) presented in Chapter 3. A hybrid nonlinear filter that consists of nonlinear and linear sub-predictors is introduced in section 8.5. Finally, the concluding remark is presented in the last section of this chapter.

8.2 A New Concurrent Algorithm For Adaptive Filtering In Parallel Signal Processing

This section introduces *concurrency* in *Lyapunov theory-based adaptive filtering* (CLAF) algorithm for adaptive filtering. The proposed super filter consists of numbers of sub-filters. The sum of all sub-filter output forms the output of the super filter. We refer to this algorithm as CLAF algorithm because those sub-filters can be run in parallel. This implementation will be particularly useful in the real-time implementation of large order filter when the computational time per iteration is critical and the LAF, RLS (recursive least square) and LMS (least mean square) algorithms are not suitable. This scheme operates as follow: a Lyapunov function is first defined for the error between the super filter output and the desired response. Those sub-filters weight parameters are adaptively adjusted by the CLAF algorithm so that the error converges to zero asymptotically.

The emergence of stable *Lyapunov theory-based adaptive filtering* (LAF) algorithm in Chapter 3, has been a major development in the design of adaptive algorithm. The LAF has provided good performance for a computational cost that is of the same order as *stochastic gradient* (SG) or *fast least square* (FRLS) algorithms [1]-[3] that lie in the range of few *N*s approximately, where *N* is the filter length. Furthermore, LAF also provides better stability and convergence properties. Hence it is necessitated to further progress in the LAF design. When the order of the filter is few hundreds or thousands, such as for echo cancellation in audio conferencing, implementation of FRLS may be infeasible in real-time applications especially for nonlinear filters. FRLS has higher computational complexity, while SG is not suited because of the slow convergence rate. LAF might provide a solution to the convergence and stability problems, but not the computational time saving when a very large filter order is implemented. Therefore the exploitation of concurrency in corresponding to the saving in the computational time required per iteration is necessary. Therefore these reasons lead to the proposed CLAF algorithm.

8.2.1 Concurrent Lyapunov Theory-Based Adaptive Filtering (CLAF)

Figure 8.1 illustrates the adaptive filter using CLAF algorithm. The filter is referred as a super filter and is decomposed into q sub-filters by partitioning $X(k)$ and $W(k)$ into q vectors each as

$$X^T(k)=[X_1(k), X_2(k), \ldots, X_i(k), \ldots, X_q(k)]^T \tag{8.1}$$

$$W^T(k)=[W_1(k), W_2(k), \ldots, W_i(k), \ldots, W_q(k)]^T \tag{8.2}$$

$X_i(k)$ and $W_i(k)$ are the data vector and weight vector of the ith sub-filter, respectively. The dimension of both $X_i(k)$ and is $W_i(k)$ $N_i \times 1$ such that $\sum_{i=1}^{q} N_i = N$.

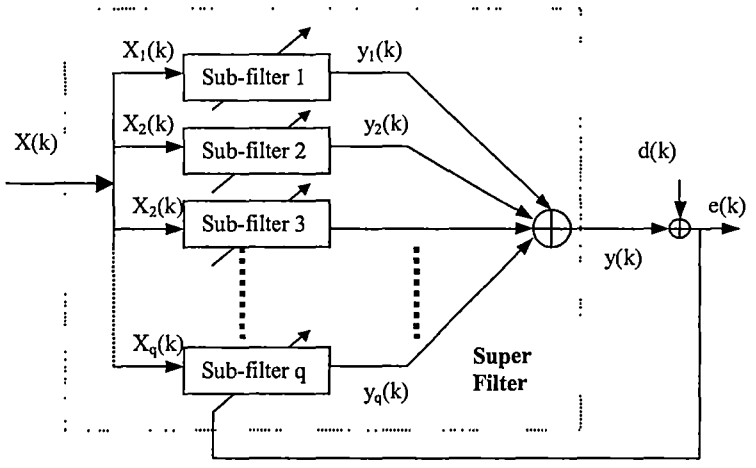


Figure 8.1: The proposed super filter with CLAF algorithm

$W_i(k)$ for the ith sub-filters, are computed independently and concurrently using their respective data vectors. The CLAF is applied to the filter to adaptively update the filter coefficients so that the error converges to zero asymptotically. The desired response can be expressed as follow:

$$d(k) = \sum_{i=1}^{q} d_i(k) \tag{8.3}$$

where $d_i(k)$ represents the component of $d(k)$ that is contributed by the ith subsystem. In this case, $d_i(k) = d(k)$ for all $1 \leq i \leq q$ is assumed. If we know the $d_i(k)$'s, we can

easily implement the super filter as q sub-filters, each operating in parallel and independent of each other. The output $y(k)$ of the super filter is computed by summing the outputs of the sub-filters.

$$y(k) = \sum_{i=1}^{q} y_i(k) \tag{8.4}$$

The sub-filters' parameters are adjusted based on the error, $e(k)$. In other words, we assume $e_i(k) = e(k)$ and $d_i(k) = d(k)$ for all $1 \leq i \leq q$. The sub-filters' coefficient vectors are updated using the following equation:

$$W_i(k) = W_i(k-1) + K_i(k)\alpha(k) \tag{8.5}$$

where $$\alpha(k) = d(k) - \sum_{i=1}^{q} W_i(k-1)X_i(k) \tag{8.6}$$

$$K_i(k) = \frac{X_i(k)}{\| X_i(k) \|^2}\left(1 - \beta_i \frac{|e(k-1)|}{|\alpha(k)|}\right) \tag{8.7}$$

or $$K_i(k) = \frac{X_i(k)}{\lambda_1 + \| X_i(k) \|^2}\left(1 - \beta_i \frac{|e(k-1)|}{\lambda_2 + |\alpha(k)|}\right) \tag{8.8}$$

where λ_1, λ_2 are small positive numbers and the sum of $\beta_i(k)$ must be less 1 or $0 \leq \sum_{i=1}^{q} \beta_i < 1$.

8.2.2 The design of the CLAF using Lyapunov theory

The design of the CLAF can be described by the following theorem:

Theorem 8.1: For the given desired response $d(k)$, if the weight vectors $W_i(k)$ of the super-filter $y(k) = \sum_{i=1}^{q} W_i(k)X_i(k)$ is updated as follows

$$W_i(k) = W_i(k-1) + K_i(k)\alpha(k)$$

and $$K_i(k) = \frac{X_i(k)}{\| X_i(k) \|^2}\left(1 - \beta_i \frac{|e(k-1)|}{|\alpha(k)|}\right) \tag{8.9}$$

where $0 \leq \sum_{i=1}^{q} \beta_i < 1$, then the error, $e(k)$ asymptotically converges to zero.

Proof: Define a Lyapunov function

$$V(k) = e^2(k) \tag{8.10}$$

Then, $\Delta V(k) = V(k) - V(k-1) = e^2(k) - e^2(k-1)$

$$\begin{aligned}
&= (d(k) - \sum_{i=1}^{q} W_i^T(k) X_i(k))^2 - e^2(k-1) \\
&= (d(k) - \sum_{i=1}^{q} (W_i^T(k-1) + K_i(k)\alpha(k)) X_i(k))^2 - e^2(k-1) \\
&= (d(k) - \sum_{i-1}^{q} W_i^T(k-1) X_i(k) - \sum_{i-1}^{q} K_i(k)\alpha(k) X_i(k))^2 - e^2(k-1) \\
&= (\alpha(k) - \sum_{i-1}^{q} K_i(k)\alpha(k) X_i(k))^2 - e^2(k-1) \\
&= \alpha^2(k)\left(1 - \sum_{i-1}^{q} K_i(k) X_i(k)\right)^2 - e^2(k-1)
\end{aligned} \tag{8.11}$$

Using the expression (8.9) in the expression (8.11), we have

$$\Delta V(k) = -(1 - \sum_{i-1}^{q} \beta_i^2) e^2(k-1) < 0 \tag{8.12}$$

Remark 8.1: The design, stability analysis of the error dynamics and convergence analysis are similar as those in Chapter 3 if we replace κ and $g(k)$ in Chapter 3 by $0 \le \sum_{i=1}^{q} \beta_i < 1$ and K_i respectively.

8.2.3 Saving In Computation Time

Before proceeding to the computational time required per iteration by CLAF, it is worthwhile to examine the same for LMS and RLS if their inherent concurrency is exploited. Authors in [85] have proposed a new family of concurrent algorithms for adaptive filter, these include PLMS and PRLS [85]. Here we define μ as the time required for one multiplication and neglect the time required for addition operations. For standard LMS, it requires $2N\mu$ time if we have just one processor. When the LMS algorithm is implemented concurrently, we have $N+1$ processor and allocate one processor for each coefficient and one (called EP) for computing $e(k)$. Thus we can observe from *Table 8.1* that the total time required is about 2μ + (*plus*) the time required by the processor for accessing $y_i(k)$ + the time required for global broadcast of $e(k)$ to N processor. Therefore the time required is very small and independent of

N. This strategy may not be advantageous for small values of *N*, but for high order filter it will offer significant saving.

In the same way, authors in [85] allocate one processor for each sub-filter in PRLS and exploit its natural concurrency. When PRLS is developed around RLS, as shown in *Table 8.2*, the steps 1 to step 5 can be done concurrently for each sub-filter. The time required for these steps is simply the time required by the longest or highest order sub-filter for steps 1 to 5 + time required for global broadcast of $e(k)$ + the time required by the EP to access $y_i(k)$ + the time required for global broadcast of $e(k)$ to *N* processor. Authors in [85] mentioned that the expression for time remains unchanged except for the first term when PRLS is developed around any FLS algorithm. The time for first term is replaced by the time required by the longest sub-filter for the computation of its gain $K_i(k)$, output $y_i(k)$ and $W_i(k)$ updated. Hence the smaller value of the longest sub-filter length, the greater will be the savings in computational time compared with the time requires by even FLS algorithm with one processor, especially if *N* is large. General expression for the computational complexity is not rewritten in [85] because they depend on the chosen configuration and the version of conventional or fast LS algorithm selected for the gain computation of the sub-filter.

Table 8.3 illustrates the concurrent implement of LAF (CLAF). It is notice that the steps 1 to step 4 can be done concurrently for each sub-filter. The time required for these steps is simply the time required by the longest sub-filter for steps 1 to step 4 + time required for global broadcast of $e(k)$ + the time required by the EP to access $y_i(k)$ + the time required for global broadcast of $e(k)$ to *N* processor. If we compared *table 8.2* and *table 8.3*, less step is required for CLAF compared to PRLS. The time required for CLAF is less than that of PRLS. However, this 'less computational time' is not necessary applied to all concurrent implementation of fast versions of RLS. Therefore, CLAF has the potential of being used in real-time applications where computational time required per iteration is critical.

Table 8.1: Concurrent Implementation of The LMS Algorithm [85]

Step 1: Do for $i=1, 2, ...N$	$y_i(k)=X_i^T(k)W_i(k)$
Step 2.	$e(k)=d(k)-\sum_{i=1}^{N} y_i(k)$
Step 3. Do for $i=1, 2, ...N$	$W_i(k+1)=W_i(k)+\tau X_i(k)\, e(k)$

Table 8.2: Concurrent Implementation of The RLS Algorithm (Develop around RLS) [85]

Do steps 1 to 5 for $i=1, 2, ...q$	Step 1	$Y_i(k)=\lambda^{-1}C_i(k-1)X_i(k)$
	Step 2	$g_i(k)=\dfrac{Y_i(k)}{1+X_i^T(k)Y_i(k)}$
	Step 3	$C_i(k)=\lambda^{-1}C_i(k-1)-g_i(k)Y_i^T(k)$
	Step 4	$W_i(k)=W_i(k-1)+g_i(k)\, e(k)$
	Step 5	$y_i(k)=X_i^T(k)W_i(k)$
	Step 6	$e(k)=d(k)-\sum_{i=1}^{q} y_i(k)$

Table 8.3: Concurrent Implementation of The LAF Algorithm

Do steps 1 to 4 for $i=1, 2, . q$	Step 1	$\alpha(k)=d(k)-\sum_{i=1}^{q} W_i(k-1)X_i(k)$
	Step 2	$K_i(k)=\dfrac{X_i(k)}{\lambda_1+\|X_i(k)\|^2}\left(1-\beta_i\dfrac{\|e(k-1)\|}{\lambda_2+\|\alpha(k)\|}\right)$
	Step 3	$W_i(k)=W_i(k-1)+K_i(k)\alpha(k)$
	Step 4	$y_i(k)=X_i^T(k)W_i(k-1)$
	Step 5	$e(k)=d(k)-\sum_{i=1}^{q} y_i(k)$

8.2.4 Simulation Examples

The simulation examples are performed to demonstrate the performance of CLAF filter. The first example demonstrates CLAF's performance when the super filter input signal is corrupted by the additive random noise that is bounded. A super filter with 5 sub-filters, $q=5$ is considered. The adaptive gain is updated according to the expression (8.8). In the first case, λ_1, λ_2 and β_i in (8.8) are chosen as follow: $\lambda_1=\lambda_2=0.001$, and $\beta_1=\beta_2=...=\beta_5=0.001$. The result illustrated in *Figure 8.2a* that shows the comparison of $d(k)$ and $y(k)$. The square error, $e^2(k)$ is illustrated in *Figure 8.2b*. The performance of the CLAF filter can be further improved by properly

choosing smaller parameters λ_1, λ_2 and β_t. It has been shown in Chapter 3, the smaller value of these parameters, the faster the error convergence rate and the smaller the error is.

Simulations of the same setup with PRLS and PLMS are also accomplished for comparison. The simulation of the same number of sub-filter with PRLS algorithm is first presented. The results in *Figure 8.3a* and *Figure 8.3b* (forgetting or weighting factor, ρ= 0.2) reveal the output signal of PRLS method has higher noise level compared to that of CLAF by observing the $e^2(k)$. Thus CLAF has fast convergence speed, good tracking property and is highly stable. Simulation results for PLMS are illustrated in *Figure 8.4a* and *Figure 8.4b*.

In summary, the section 8.2 has introduced the CLAF that has provided a new and alternative approach to conventional parallel or concurrent algorithms and hopefully suggested a new research area of adaptive parallel signal processing.

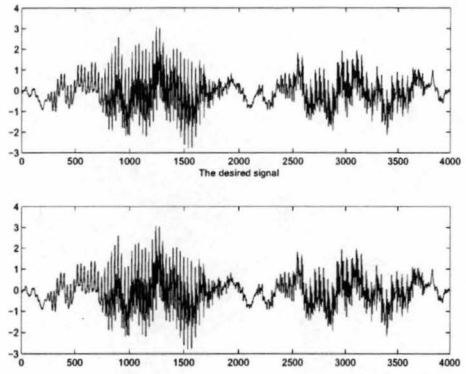


Fig. 8.2a: CLAF-Desired output, $d(k)$ & filter output, $y(k)$

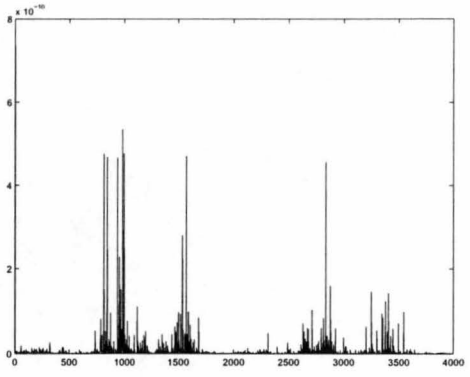


Fig. 8.2b: CLAF -The square error, $e^2(k)$

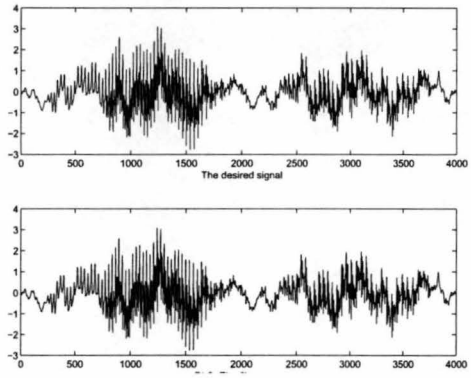


Fig. 8.3a: PRLS - Desired output,$d(k)$ & filter output, $y(k)$

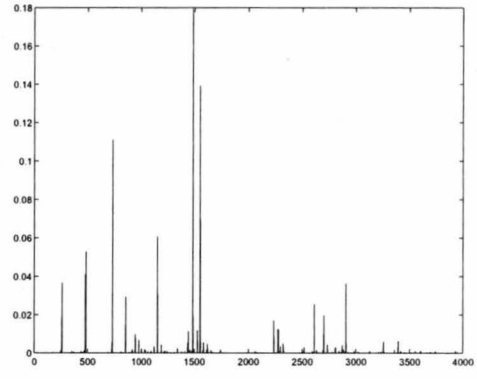


Fig. 8.3b: PRLS - The square error, $e^2(k)$

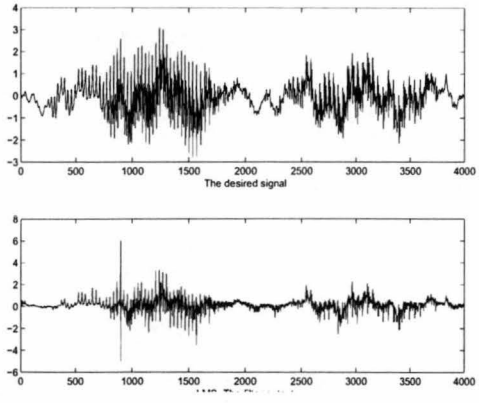


Fig. 8.4a: PLMS - Desired output, $d(k)$ & filter output, $y(k)$

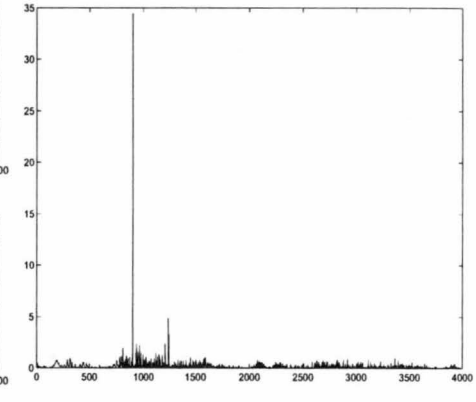


Fig. 8.4b: PLMS - The square error, $e^2(k)$

8.3 Complex-valued Lyapunov Theory-based Adaptive Filtering (Complex-LAF)

Most available adaptive filters are real-valued and are suitable for signal processing in real multi-dimensional space. In some applications, however, signals are complex-valued and processing is done in complex multi-dimensional space. An example is the channel equalization of communication channels with complex signaling schemes such as quadrature amplitude modulation (QAM). For complex signal processing problems, many existing adaptive algorithms cannot directly be applied. Although for certain applications it is possible to reformulate a complex signal processing problem so that a real-valued adaptive algorithm can be used to solve the problem, it is not always feasible to do so. Furthermore it is preferred to preserve the concise formulation and elegant structure of complex signals. [86],[87]

This section presents a complex-valued version of the LAF in Chapter 3. A Lyapunov function is first defined for the real and imaginary parts of the error between the desired response and the filter output. Filter real and imaginary coefficients are then adaptively adjusted in parallel based on Lyapunov Stability Theory so that the error can converge to zero asymptotically. Simulation examples are included to demonstrate the performance that can be achieved based on the new design.

The framework of complex adaptive Lyapunov filter problem is described as follows: the observations $\{x(k)\}$ are regarded as the complex filter inputs and $d(k)$ is the complex reference signal. The complex error $e(k)$ is denoted as:

$$\begin{aligned} e(k) &= d(k) - y(k) \\ &= \mathrm{Re}[e(k)] + j\,\mathrm{Im}[e(k)] = e_R(k) + je_I(k) \end{aligned} \tag{8.13}$$

where $y(k) = y_R(k) + \mathrm{j}\, y_R(k)$

$$= \boldsymbol{H}^T(k)\boldsymbol{X}(k) \tag{8.14}$$

Figure 8.5 shows two ways of representing the complex linear combiner. The complex input vector $\boldsymbol{X}(k)$ and complex weight vector $\boldsymbol{H}(k)$ are given by

$$\boldsymbol{H}(k) = H_R(k) + \mathrm{j}\, H_I(k) \tag{8.15}$$

and $$\boldsymbol{X}(k) = X_R(k) + \mathrm{j}\, X_I(k) \tag{8.16}$$

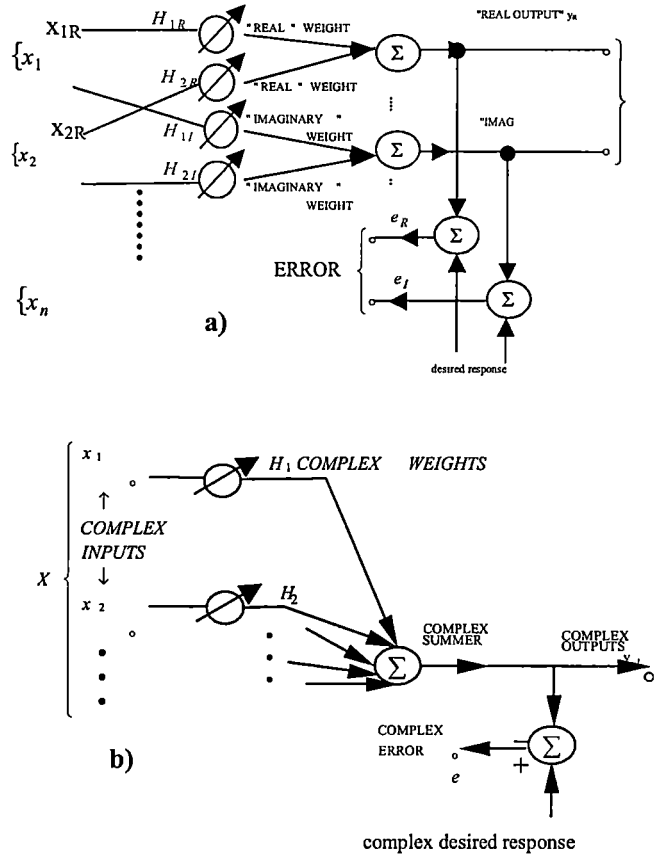


Figure 8.5: Complex adaptive linear combiner. a) In block diagram form. b) In schematic representation.

8.3.1 Complex Lyapunov Theory-based Adaptive Filtering Algorithm

The complex adaptive Lyapunov algorithm consists two parts. The first part updates the real part of filter coefficients while the second part updates the imaginary part of filter coefficients. Both coefficients are updated in parallel. The structure of this algorithm is illustrated in *Figure 8.6.*

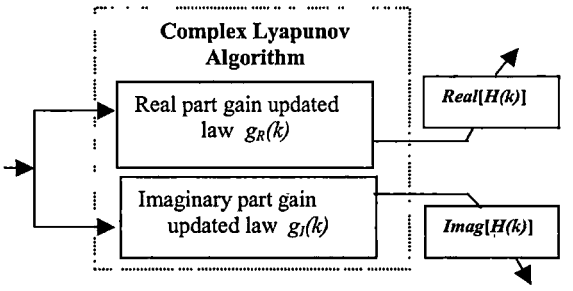


Figure 8.6: Structure of Complex LAF Algorithm

The following algorithms is used to update the filter real and complex parameters vectors:

$$Re[H(k)] = Re[H(k-1)] + g_R(k)\alpha_R(k) \tag{8.17a}$$

$$Im[H(k)] = Im[H(k-1)] + g_I(k)\alpha_I(k) \tag{8.17b}$$

$$\alpha_R(k) = Re[d(k)] - Re[H^T(k-1)]Re[X(k)] \tag{8.18a}$$

$$\alpha_I(k) = Im[d(k)] - Im[H^T(k-1)]Im[X(k)] \tag{8.18b}$$

$$g_R(k) = \frac{\mathrm{Re}[X(k)]}{\|\mathrm{Re}[X(k)]\|^2}\left(1-\kappa_1\frac{|e_R(k-1)|}{|\alpha_R(k)|}\right) \tag{8.19a}$$

$$g_I(k) = \frac{\mathrm{Im}[X(k)]}{\|\mathrm{Im}[X(k)]\|^2}\left(1-\kappa_2\frac{|e_I(k-1)|}{|\alpha_I(k)|}\right) \tag{8.19b}$$

or

$$g_R(k) = \frac{X_R(k)}{\|X_R(k)\|^2+\lambda_1}\left(1-\kappa_1\frac{|e_R(k-1)|}{\lambda_2+|\alpha_R(k)|}\right) \tag{8.20a}$$

$$g_I(k) = \frac{X_I(k)}{\|X_I(k)\|^2+\lambda_3}\left(1-\kappa_2\frac{|e_I(k-1)|}{\lambda_4+|\alpha_I(k)|}\right) \tag{8.20b}$$

where λ_1, λ_2, λ_3 ,λ_4 are small positive numbers.

Note: subscript R = real and I = imaginary

8.3.2 Design of the Complex-LAF Filter

The design of the complex adaptive filter is described by Theorem 8.2:

Theorem 8.2: For the given $d(k)$, if the filter parameters vector $\boldsymbol{H}(k)$ of the filter $y(k) = \boldsymbol{H}^T(k)\boldsymbol{X}(k)$ is updated as follows

$$Re[\mathrm{H}(k)] = Re[H(k-1)] + g_R(k)\alpha_R(k)$$

$$Im[\mathrm{H}(k)] = Im[H(k-1)] + g_I(k)\alpha_I(k)$$

and
$$g_R(k) = \frac{\mathrm{Re}[X(k)]}{\|\mathrm{Re}[X(k)]\|^2}\left(1-\kappa_1\frac{|e_R(k-1)|}{|\alpha_R(k)|}\right) \tag{8.21a}$$

$$g_I(k) = \frac{\mathrm{Im}[X(k)]}{\|\mathrm{Im}[X(k)]\|^2}\left(1-\kappa_2\frac{|e_I(k-1)|}{|\alpha_I(k)|}\right) \tag{8.21b}$$

where $0 \le \kappa_1 , \kappa_2 < 1$, then the real and imaginary parts of the error $e_R(k)$, $e_I(k)$ asymptotically converges to zero.

Proof: Define a Lyapunov function of real and imaginary parts of the error $e(k)$

$$\begin{aligned} V(k) &= e(k)e^*(k) = \mathrm{Re}[e(k)]^2 + \mathrm{Im}[e(k)]^2 \\ &= e_R(k)^2 + e_I(k)^2 \end{aligned} \tag{8.22}$$

$$\begin{aligned} \Delta V(k) &= V(k) - V(k-1) \\ &= \mathrm{Re}[e(k)]^2 + \mathrm{Im}[e(k-1)]^2 - \mathrm{Re}[e(k-1)]^2 - \mathrm{Im}[e(k-1)]^2 \\ &= e_R{}^2(k) + e_I{}^2(k) - e_R{}^2(k-1) - e_I{}^2(k-1) \\ &= (s_R(k-d) - H_R{}^T(k)X_R(k))^2 - e_R{}^2(k-1) + (s_I(k-d) - H_I{}^T(k)X_I(k))^2 - e_I{}^2(k-1) \\ &= (s_R(k-d) - H_R{}^T(k)X_R(k))^2 - e_R{}^2(k-1) \\ &\quad + (s_I(k-d) - H_I{}^T(k)X_I(k))^2 - e_I{}^2(k-1) \\ &= (s_R(k-d) - (H_R{}^T(k-1) + g_R{}^T(k)\alpha_R(k))X_R(k))^2 \\ &\quad - e_R{}^2(k-1) + (s_I(k-d) - (H_I{}^T(k-1) + g_I{}^T(k)\alpha_I(k))X_R(k))^2 - e_I{}^2(k-1) \end{aligned}$$

$$
\begin{aligned}
&= (s_R(k-d) - (H_R^T(k-1) + g_R^T(k)\alpha_R(k))X_R(k))^2 - e_R^2(k-1) \\
&\quad + (s_I(k-d) - (H_I^T(k-1) + g_I^T(k)\alpha_I(k))X_R(k))^2 - e_I^2(k-1) \\
&= (s_R(k-d) - H_R^T(k-1)X_R(k) - g_R^T(k)\alpha_R(k)X_R(k))^2 - e_R^2(k-1) \\
&\quad + (s_I(k-d) - H_I^T(k-1)X_I(k) - g_I^T(k)\alpha_I(k)X_I(k))^2 - e_I^2(k-1) \\
&= (s_R(k-d) - H_R^T(k-1)X_R(k) - g_R^T(k)\alpha_R(k)X_R(k))^2 - e_R^2(k-1) \\
&\quad + (s_I(k-d) - H_I^T(k-1)X_I(k) - g_I^T(k)\alpha_I(k)X_I(k))^2 - e_I^2(k-1) \\
&= (\alpha_R(k) - g_R(k)\alpha_R(k)X_R(k))^2 - e_R^2(k-1) \\
&\quad + (\alpha_I(k) - g_I(k)\alpha_I(k)X_I(k))^2 - e_I^2(k-1) \\
&= \alpha_R^2(k)\left(1 - g_R^T(k)X_R(k)\right)^2 - e_R^2(k-1) + \alpha_I^2(k)\left(1 - g_I^T(k)X_I(k)\right)^2 - e_I^2(k-1)
\end{aligned}
\tag{8.23}
$$

Note: subscript R = real and I = imaginary

Using the expression (8.21a,b) in expression (8.23), we have

$$\Delta V(k) = -(1-\kappa_1^2)e_R^2(k-1) - (1-\kappa_2^2)e_I^2(k-1) < 0 \tag{8.24}$$

Remark 8.2: The error $e_R(k)$ will not converge to zero if the adaptive gains $g_R(k)$ and $g_I(k)$ are adjusted using expressions (4.7a, b). However, the $e_R(k)$ will converge to a ball centred at the origin of the real error space with radius of the ball depends on λ_1 and λ_2 values, while the $e_I(k)$ will converge to a ball centred at the origin of the imaginary error space with radius of the ball relies on λ_3 and λ_4 values. As explained in Chapter 3, smaller these constant values contribute smaller errors, $e_R(k)$ and $e_I(k)$.

8.3.3. Simulation Examples

The performance of the proposed complex-LAF filter is illustrated using a complex-valued nonlinear communications channel. The transmitted signal $s(k)=s_R(k)+js_I(k)$ and the additive noise $n(k)=n_R(k)+jn_I(k)$, are complex. The nonlinear element is defined by

$$u(k) = \frac{2s(k)}{1+|s(k)|^2}\exp\left(j\frac{\pi}{3}\frac{|s(k)|^2}{1+|s(k)|^2}\right) \tag{8.25}$$

This static nonlinearity is used to represent the nonlinear high power amplifier in the transmitter [86],[87]. Therefore this channel is characterized by nonlinear model. The time dispersive transmission medium is usually modeled as a *finite impulse response* (FIR) filter with a transfer function

$$A(z) = (-1.0119 + j*0.7589) + (-0.3796 + j*0.5059)^{-1} + (-1 + j*0.5000)z^{-2} \qquad (8.26)$$

The first simulation is to show the robustness of the complex filter to additive noise. where both $n_R(k)$ and $n_I(k)$ are bounded additive noise: $0 < n_R(k), n_I(k) < 0.4$ that gives SNR ≈18. *Figure 8.7a* illustrates filter input signal, $Re[x(k)]$, $Im[x(k)]$, that is the channel output corrupted by additive noise. *Figure 8.7b* reveals the comparison of the desired signal, $Re[d(k)] = \mathrm{Re}[s(k-1)]$, $Im[d(k)]$ and the filter output $Re[y(k)]$ and $Im[y(k)]$ respectively. It can be seen that the complex-LAF filter is highly suitable for nonlinear channel equalization.

The second simulation is presented to reveal the robustness of the filter to an unexpected large disturbance. The disturbance is indicated in *Figure 8.8a*. *Figure 8.8b* shows the comparison of the filter output $y(k)$ and the desired response, $d(k)$ for real and imaginary parts respectively. Effects of the additive complex noise and the disturbance are greatly reduced. $Re[y(k)]$, $Im[y(k)]$ follow the desired response, $Re[d(k)]$, $Im[d(k)]$ closely. Hence the proposed filter is robust to the additive noise and disturbance.

In conclusion, the section 8.3 has provided a new approach in designing a complex adaptive filter using the Lyapunov Stability Theory. Simulation has revealed good error convergence, robustness to additive noise and large disturbance presented in the channel equalization.

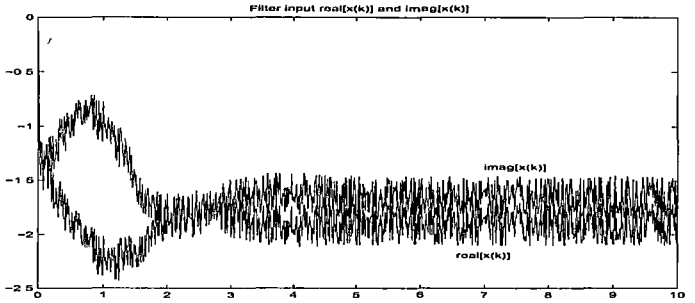


Figure 8.7a: Filter input signal, $Re[x(k)]$, $Im[x(k)]$

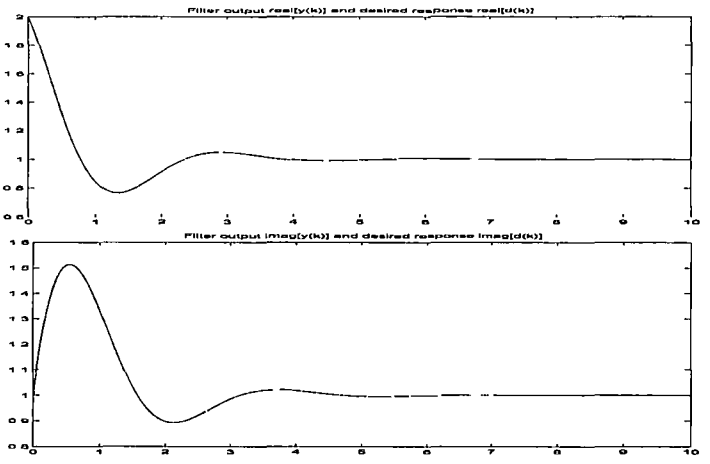


Figure 8.7b: Filter output & desired signal *Re[y(k)]*, *Im[y(k)]*,*Re[d(k)]*, *Im[d(k)]*

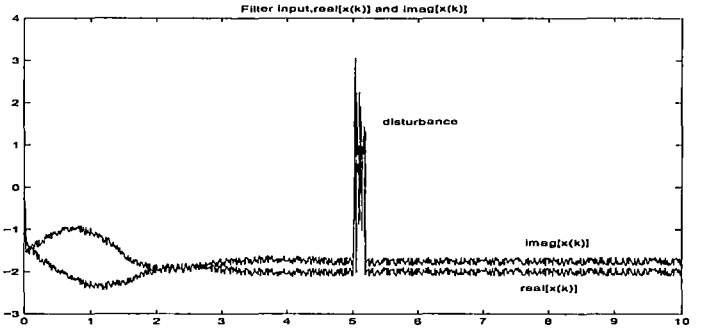


Figure 8.8a: Filter input signal, *Re[x(k)]*, *Im[x(k)]*

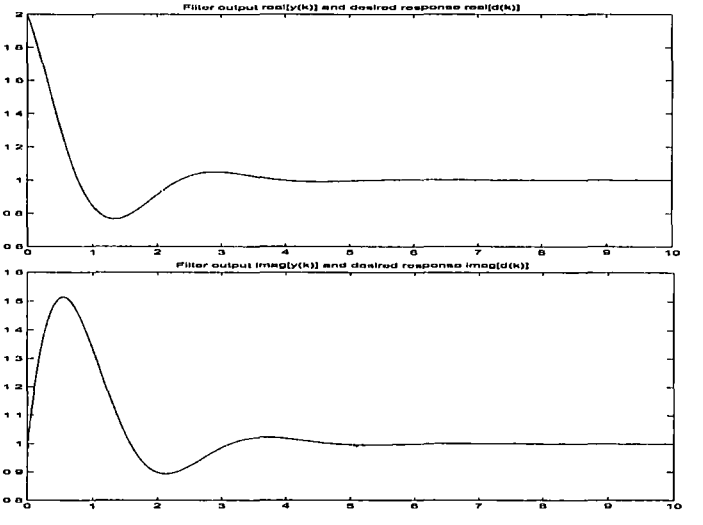


Figure 8.8b: Filter output & desired signal *Re[y(k)]*, *Im[y(k)]*,*Re[d(k)]*, *Im[d(k)]*

8.4 A New Approach In Feedforward Active Noise Control Using Lyapunov Stability Theory

In this section, two new and efficient algorithms for active noise control (ANC) system are proposed. The conventional *Filtered-X LMS* (least mean square) and *Filtered-U LMS* will be introduced briefly in the initial part of the section. This is followed by the new implementation of the proposed *Filtered-X Lyapunov theory-based* (FXLYP) and *Filtered-U Lyapunov theory-based* algorithms for ANC. Like FXLMS, these algorithms have included the secondary path effect. The secondary path effect is adaptively estimated by a new proposed overall online modeling technique that employs *Lyapunov theory-based adaptive filtering* (LAF) in Chapter 3. Simulation examples are performed to demonstrate the performance of this scheme.

8.4.1 Active Noise Control

Noise control has become ever more important in recent years. Interest in active methods for the suppression of noise and vibration has grown recently, as evidenced by the numerous review articles and books that have appeared on the subject [88]-[95]. Although the potential for active noise and vibration control has long been recognized [96], successful implementations of these technique have begun to appear maturation of technology in three areas: 1) novel electroacoustic transducers, 2) advanced adaptive control algorithm, and 3) inexpensive and reliable digital signal processing (DSP) hardware. As advances in these areas are developed, active suppression of noise and vibration can be expected to find wider use in a number of commercial, industrial, and military applications. Specifically, ANC [88],[91] has been successfully applied to HVAC (Heating, ventilating and air conditioning) systems [89], exhaust noise and motor noise [89]. Furthermore, national and multinational programs and policies are being established to reduce and control environmental noise.

In general, ANC is based on the principle of the destructive interference between a primary noise source and a secondary source, whose acoustic output is governed by a

controller. [88],[92] The output of the secondary source (i.e. loudspeaker) has to be in exact anti-phase with the acoustic wave produced by the primary noise source. A typical ANC system in a duct is shown in *Figure 8.9*. There are two distinct strategies for ANC: Feedforward and feedback. The formal strategy is the one we considered in this paper. For feedforward control, the noise from the primary source travels, from left to right, as plane waves through the dust. A microphone located upstream from the secondary source detects the incident noise waves and supplies the controller with an input signal. The controller sends a signal to the secondary source (i.e. loudspeaker) which is in anti-phase with the disturbance. A microphone located downstream picks up the residuals and supplies the controller with an error signal.

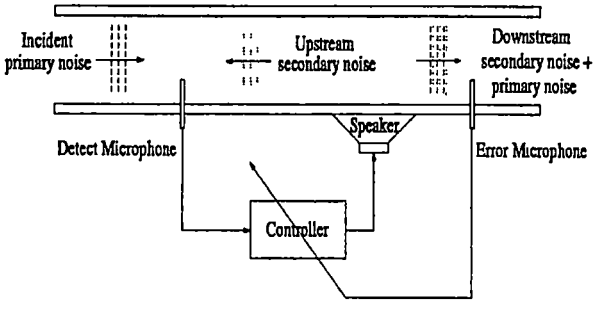


Figure 8.9: Single-channel broadband feedforward ANC system in a duct

The most popular adaptive algorithm for active noise and vibration control is the *Filtered-X LMS* (FXLMS) algorithm [88],[97]-[104]. Its equivalent block diagram for single channel ANC using FXLMS is illustrated in *Figure 8.10*. This algorithm is a modification of the well known LMS algorithm, in which the reference signal is filtered to compensate for a filtering operation inherent to the adaptation loop. The introduction of the secondary path and the filtered reference signal in the system complicate significantly the analysis of the adaptive algorithm behavior. Analysis results derived for the conventional LMS algorithm do not apply to the filtered case. Also, simplifying assumptions that facilitate the analysis of the LMS algorithm cannot be easily extended to the FXLMS algorithm. This is the case of the independence theory. The signal correlations introduced by the filtering operations render the independence theory inadequate for the statistical analysis of the algorithm

behavior. This is a substantial drawback, since the exact analysis without the independence assumption becomes very cumbersome, even for the much simpler case of the conventional LMS algorithm. Most of the stochastic analyses of the FXLMS algorithm available in the literature concentrate on the stability limits of the algorithm. [88],[97] These results are important for the proper design of the algorithm. However, a more complete analytical model is necessary for a better understanding of the algorithm's properties, including transient and steady-state behavior under different implementation conditions.

In order to consider the feedback dynamics as a part of the overall plant in ANC, the *Filtered-U LMS* (FULMS) [88],[97] algorithm is derived. *Figure 8.11* illustrates its equivalent block diagram. The simplifying assumptions in the derivation of these algorithms make them conservative in terms of reducing the admissible update gain factors. Consequently, their convergence performance is unsatisfactory. Additionally, since the error microphone should be located far from the secondary source to avoid the near-field effects of sound, there is a time delay in the secondary path dynamics. This represents yet another source of deterioration of the convergence behavior. Several modified algorithms have been proposed to improve the convergence behavior [88],[105]. These modifications are made either by considering the effect of time delay [88] or by directly applying the Lyapunov theory in the derivation of the algorithms [88],[20]. Both approaches yield the same solution, which here is referred to as stable adaptive algorithm [88]. However, it still does not guarantee stability in the presence of model uncertainty and disturbance. Hence, a robust stable algorithm is desired for ANC that includes the feedback dynamic.

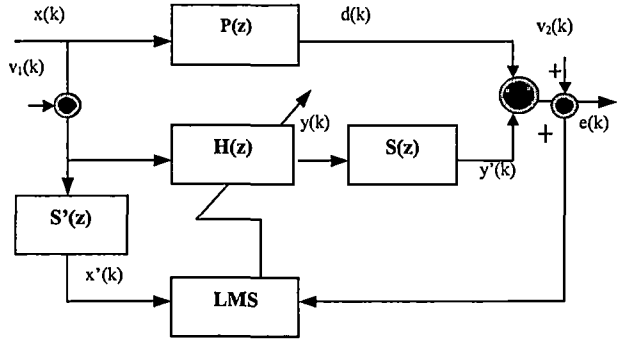


Figure 8.10: ANC using FXLMS algorithm

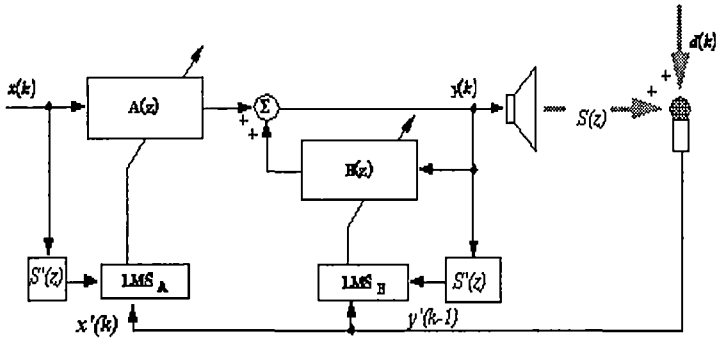


Figure 8.11.: ANC using FULMS algorithm

In general, the FXLMS algorithm can be summarized as follow:

$$h_i(k+1) = h_i(k) - \mu(k)e(k)x'(k-i) \tag{8.27}$$

where $\mu(k)$ is the algorithm step size or learning rate at time k, the filtered input sequence $x'(k)$ is computed as

$$x'(k) = \sum_{m=1}^{M} s_m(k)x(k-m) \tag{8.28}$$

and M is the FIR filter length of an appropriate estimate of the plant impulse response. In practice, the values of s_m in (8.24) and are usually obtained in a separate estimation procedure that is performed prior to the application of control. They are also estimated through the on-line modeling.

When the effects of feedback are included in ANC, the FULMS is used. Going through the derivation of the IIR-LMS algorithm, taking account of the presence of $S(z)$ results in the following FULMS weight update equations:

$$\boldsymbol{a}(k+1) = \boldsymbol{a}(k) + \mu\mathbf{x}'(k)e(k) \tag{8.29}$$

and

$$\boldsymbol{b}(k+1) = \boldsymbol{b}(k) + \mu\mathbf{y}'(k-1)e(k) \tag{8.30}$$

where $\mathbf{x}'(k) = s'(k)*\mathbf{x}(k-1)$ and $\mathbf{y}'(k-1) = s'(k)*\mathbf{y}(k-1)$.

The output of the ANC controller is given by

$$y(k) = H^T(k)U(k) \tag{8.31}$$

where $H(k) = [a(k)\ b(k)]T$, $U(k) = [x(k)\ y(k-1)]^T$.

Although FXLMS and FULMS offer simple update strategies, they require knowledge about the statistics of the input data in order to choose the proper step size, especially when on-line secondary path identification is employed. To ensure convergence, the step size is typically chosen to be smaller, causing the system to converge slowly and to exhibit poor performance.

8.4.2 New Implementation: Filtered-X Lyapunov Theory-Based Algorithm

Chapters 3-7 have proposed a range of Lyapunov theory-based algorithms. These ideas can be extended to design an adaptive algorithm in ANC controller. However, those results derived in Chapter 3 do not apply to the filtered case in ANC. Hence, a modified of LAF algorithm for ANC is proposed and it is called *Filtered-X Lyapunov theory-based algorithm* (FXLYP). To include the feedback dynamics as a part of the overall plant, a new *Filtered-U Lyapunov theory-based algorithm* (FULYP) algorithm is then derived in the section 8.4.3.

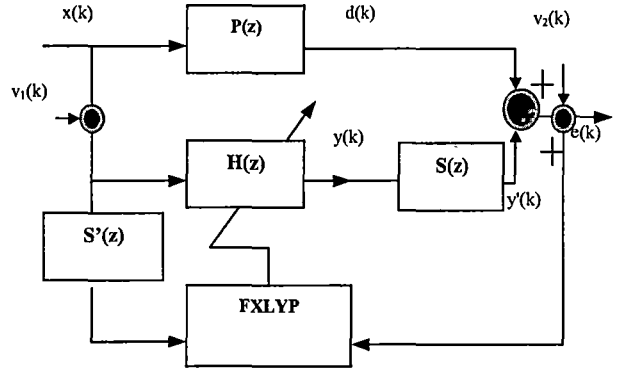


Figure 8.12: ANC with FXLYP algorithm

If the FIR filter-based controller is implemented, the output of the adaptive controller filter is

$$y(k) = \sum_{i=0}^{N-1} h_i(k)x(k-i) = H^T(k)X(k) \tag{8.32}$$

where $H(k) = [h_k(0), h_k(1), ..., h_k(N-1)]^T$, $X(k) = [x(k), x(k-1), ..., x(k-N+1)]^T$

An error sensor measures the error signal as modeled by the equation

$$e(k) = d(k) + s(k) * y(k) = d(k) + s(k) * H^T(k)X(k)$$
$$\approx d(k) + H^T(k)X'(k) \tag{8.33}$$

where $X'(k) = [x'(k), x'(k-1), ..., x'(k-N+1)]^T$ is the filtered reference signal vector with elements $x'(k) = s'(k) * x(k)$ and $s'(k)$ is the impulse response of the secondary path $S'(z)$.

Now the LAF algorithm cannot be employed directly to ANC and it has been modified as follow to suit to the ANC scheme:

$$H(k) = H(k-1) - g(k)\alpha(k) \tag{8.34}$$

$$\alpha(k) = d(k) + H^T(k-1)X'(k) \tag{8.35}$$

$$g(k) = \frac{X'(k)}{\| X'(k) \|^2}\left(1 - \kappa \frac{|e(k-1)|}{|\alpha(k)|}\right) \tag{8.36}$$

or
$$g(k) = \frac{X'(k)}{\lambda_1 + \| X'(k) \|^2}\left(1 - \kappa \frac{|e(k-1)|}{\lambda_2 + |\alpha(k)|}\right) \tag{8.37}$$

where λ_1, λ_2 are small positive numbers and $0 \leq \kappa < 1$.

Remark 8.3: The secondary path model in the ANC can be obtained by offline or online modeling techniques. One of these techniques is *overall online modeling* [88] that has the capability to model the secondary path without using an additional excitation signal. It also introduces another adaptive filter to model $P(z)$. From the expression (8.35) it is noticeable that the undesired signal, $d(k)$ is needed for the training algorithm. If the undesired signal is not available, an estimated $d(k)$ from the overall online modeling can used for the FXLYP. A new overall online modeling scheme based on the Lyapunov theory is presented in the later section so that it can be used in conjunction with the FXLYP or FULYP to have excellent performance.

8.4.3 New Overall Online Modeling Using Lyapunov Stability Theory

Current online adaptation techniques for ANC system can be divided into two classes, namely techniques that estimate the secondary path S(z) by using additive noise and overall modeling techniques without the use of additive noise. The overall modeling is the one we consider.

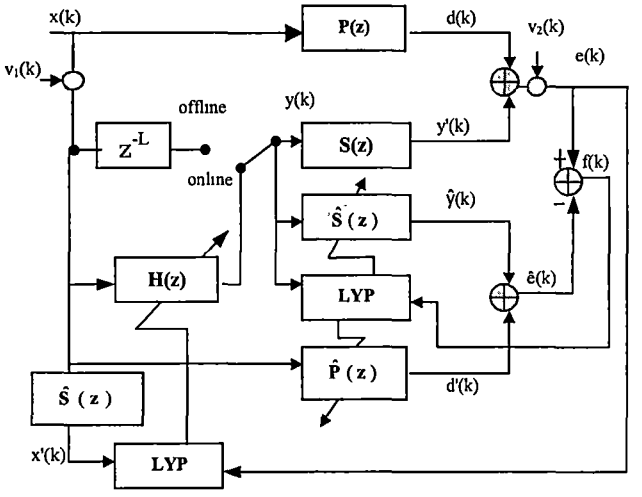


Figure 8.13: ANC system using overall modeling technique

The residual error signal *e(k)* in *Figure 8.13* can be expressed as

$$e(k) = y'(k) + d(k) = \boldsymbol{p}^T(k)\boldsymbol{x}(n) + \boldsymbol{s}^T(k)\boldsymbol{y}(k) \tag{8.38}$$

where $\boldsymbol{p}$(k) and $\boldsymbol{s}$(k) are the impulse response of P(z) and S(z) at time k respectively. On the other hand, the combined output of the adaptive filter $\hat{S}(z)$ and $\hat{P}(z)$ is

$$\hat{e}(k) = \hat{\boldsymbol{p}}(k)\boldsymbol{x}(k) + \hat{\boldsymbol{s}}(k)\boldsymbol{y}(k) = W^T(k)U^*(k) \tag{8.39}$$

where the signals vector is expressed as $U^*(k) = [\boldsymbol{x}(k)\ \boldsymbol{y}(k)]^T$, and the weights vector is $W(k) = [\hat{\boldsymbol{p}}\ (k)\ \hat{\boldsymbol{s}}\ (k)]^T$.

In this system identification configuration, *W(k)* is adjusted to approximate *S(z)* and *P(z)*. Now the LAF algorithm presented in Chapter 3 can be applied in this modeling scheme:

$$W(k) = W(k-1) + g^*(k)\alpha^*(k) \tag{8.40}$$

$$\alpha^*(k) = e(k) + W^T(k\text{-}1)U(k) \tag{8.41}$$

$$g^*(k) = \frac{U^*(k)}{\| U^*(k) \|^2}\left(1 - \beta \frac{|f(k-1)|}{|\alpha^*(k)|}\right) \tag{8.42}$$

or $$g^*(k) = \frac{U^*(k)}{\gamma_1 + \| U^*(k) \|^2}\left(1 - \beta \frac{|f(k-1)|}{\gamma_2 + |\alpha^*(k)|}\right) \tag{8.43}$$

where γ_1, γ_2 are small positive numbers and $0 \leq \beta < 1$.

Remark 8.4: The estimated $\hat{\boldsymbol{s}}(k)$ from $W(k) = [\hat{p}(k)\ \hat{\boldsymbol{s}}(k)]^T$ is then applied to the FXLYP algorithm. The estimated undesired signal, $d'(k) = \hat{p}(k)x(k)$ is used for the FXLYP if the primary undesired signal, $d(k)$ is not obtainable. Again, the designed overall modeling algorithm preserves the same characteristics or advantages of LAF and FXLYP as discussion in Chapter 3 and previous section.

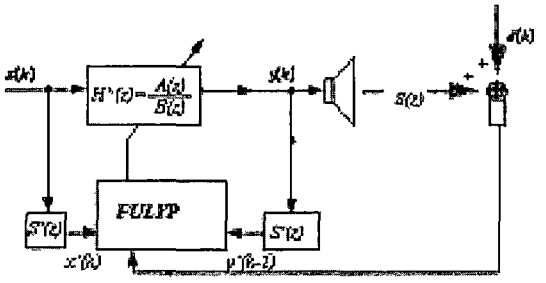


Figure 8.14: ANC using FULYP algorithm

8.4.4 New Implementation- Filtered-U LYP Algorithm

Figure 8.14 illustrates the ANC using the Filtered-U Lyapunov stability theory-based (FULYP) algorithm. If the IIR filter-based controller is implemented, the output of the adaptive controller filter is

$$\begin{aligned} y(k) &= \sum_{i=0}^{N-1} b_i(k)x(k-i) + \sum_{i=1}^{N-1} a_i(k)y(k-i) \\ &= B^T(k)X(k) + A^T(k)Y(k-1) = H^{*T}(k)X^*(k) \end{aligned} \tag{8.44}$$

where $H^*(k) = [\boldsymbol{a}(k)\ \boldsymbol{b}(k)]^T = [b_0(k), b_1(k), \cdots, b_{N-1}(k), a_1(k), a_2(k), \cdots, a_{N-1}(k)]^T$,

$$U(k) = [x(k)\ y(k-1)]^T = [x(k), x(k-1), \cdots, x(k-N+1), y(k-1), y(k-2), \cdots, y(k-N+1)]^T$$

An error sensor measures the error signal as modeled by the equation

$$e(k) = d(k) + s(k) * y(k) = d(k) + s(k) * (B^T(k)X(k) + A^T(k)Y(k-1))$$

$$\approx d(k) + H^{*T}(k)U'(k) \tag{8.45}$$

where $U'(k) = [x'(k), x'(k-1), \ldots, x'(k-N+1), y'(k-1), y'(k-2), \ldots, y'(k-N+1)]^T$ is the filtered reference signal vector with elements $x'(k) = s'(k) * x(k)$ and $s'(k)$ is the impulse response of the estimated secondary path. On the other hand, $y'(k-1) = s'(k) * y(k-1)$.

Now, the coefficient vector updated law can be summarized as

$$H^*(k) = H^*(k-1) - g(k)\alpha(k) \tag{8.46}$$

$$\alpha(k) = d(k) + H^{*T}(k-1)U'(k) \tag{8.47}$$

$$g(k) = \frac{U'(k)}{\| U'(k) \|^2} \left(1 - \kappa \frac{|e(k-1)|}{|\alpha(k)|}\right) \tag{8.48}$$

or

$$g(k) = \frac{U'(k)}{\lambda_1 + \| U'(k) \|^2} \left(1 - \kappa \frac{|e(k-1)|}{\lambda_2 + |\alpha(k)|}\right) \tag{8.49}$$

where λ_1, λ_2 are small positive numbers and $0 \leq \kappa < 1$.

Remark 8.5: Again, it is suggested that the combination of FULYP and the on-line estimation of modeling the secondary paths transfer function $S(z)$ and the primary plant $P(z)$ proposed in the section 8.4.4 can give better performance.

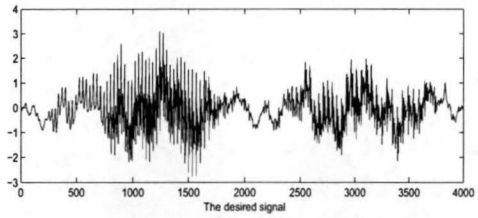
8.4.5 Simulation Examples

Simulation examples illustrate the performance of the proposed FXLYP and FULYP. Segments of broadband noises used are sampled and then are applied to the input of the ANC system. The secondary path model used is $S(z^{-1}) = z^{-2}(1 - 2z^{-2})$ which is a delayed bandpass and has zero outside the unit circle. Effects of measurement noises, $u(k)$ shown in *Figure 8.10* is also considered. The additive noise is white normal

random noise {0 1}. To compare the performance of the proposed scheme, simulation with FXLMS algorithm is also presented.

ANC controller (FIR) with FXLYP vs FXLMS - The output, *y'(k)* of the ANC with FXLYP including the controller filter and secondary path transfer function is illustrated in *Figure 8.15a*. The residual error, *e(k)* is revealed in *Figure 8.15b*. For the same setup, simulation results for FXLMS including the secondary path estimation are illustrated in *Figures 8.15c* and *8.15d*. Plots of the error, *e(k)* for FXLYP and FXLMS without additive noise are shown in *Figures 8.16a* and *8.16b* Their weights are revealed in *Figures 8.16c* and *8.16d* respectively. It is notices that the proposed controller with FXLYP can tolerate the additive noise, $v_1(k)$ and perform better than FXLMS. Thus the theoretical and simulation results have shown this scheme has given good performance.

ANC controller (IIR) with FULYP vs FULM S - *Figure 8.17a* show the output of the ANC with FULYP. The residual error, *e(k)* is revealed in *Figure 8.17b*. Those simulation results for FULMS including the secondary path estimation are illustrated in *Figure 8.17c* and *8.17d* for comparison. Controller weights with FULYP and FULMS are plotted in *Figure 8.17e* and *8.17f* when additive noise is absent. From these simulation results, the proposed FULYP is guaranteed to perform better than FULMS for ANC with IIR structure.



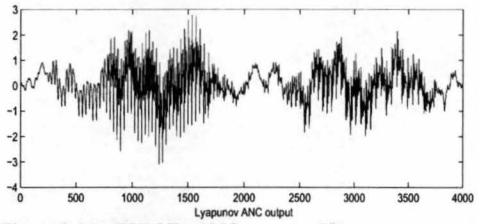
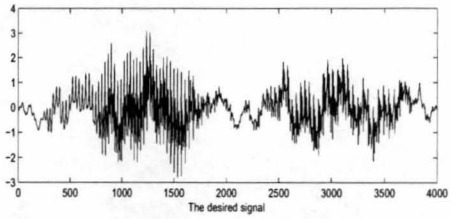


Figure 8.15a: FXLYP- ANC output, *y'(k)* & undesired signal, *d(k)*



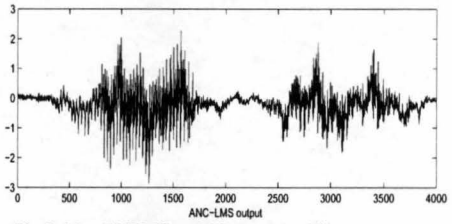


Fig 8.15c: FXLMS- ANC output, *y'(k)* & undesired signal, *d(k)*

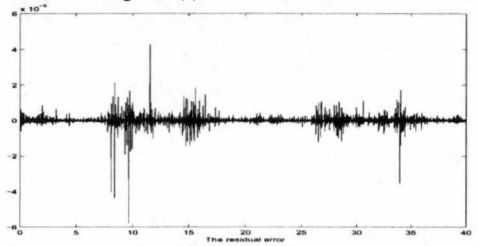


Figure 8.15b: FXLYP-The residual error, *e(k) y-axis: x10^{-6} x-axis: 4000 iterations* (with measurement noise, $v_1(k)$)

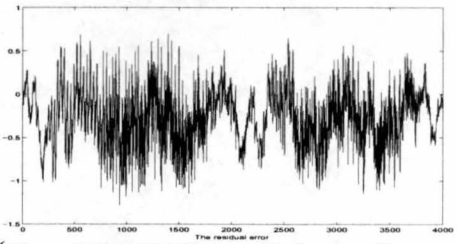


Figure 8.15d: FXLMS, the residual error, *e(k)* (with measurement noise, $v_1(k)$)

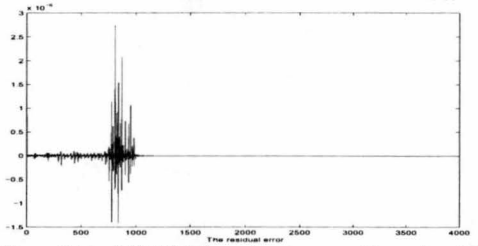


Figure 8.16a: FXLYP-The residual error, *e(k) y-axis: x10^{-6} x-axis: 4000 iterations* (without measurement noise, $v_1(k)$)

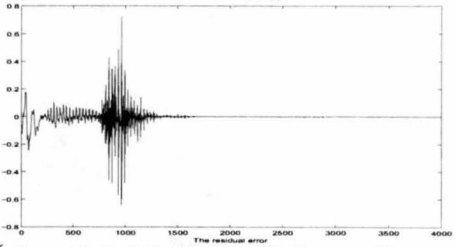


Figure 8.16b: FXLMS, the residual error, *e(k)* (without measurement noise, $v_1(k)$)

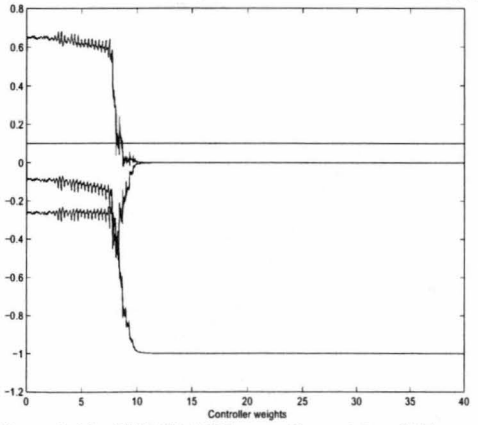


Figure 8.16c: FXLYP-ANC controller weights, *H(k)* (without measurement noise, $v_1(k)$), *4000 iterations*

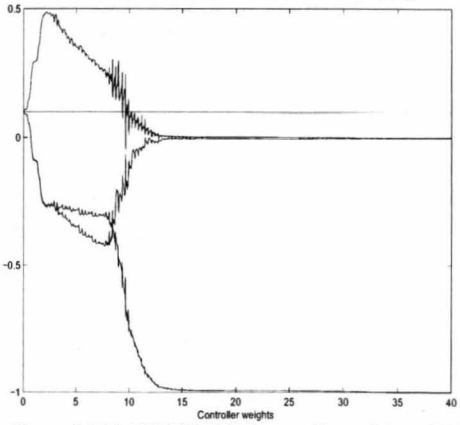
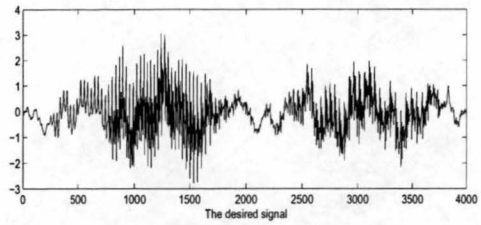


Figure 8.16d: FXLMS- ANC controller weights, *H(k)* (without measurement noise, $v_1(k)$), *4000 iterations*



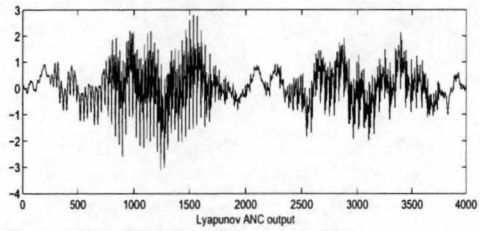


Figure 8.17a: FULYP- ANC output, *y'(k)* & undesired signal, *d(k)*

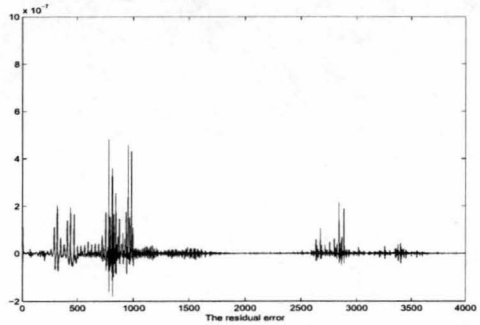


Figure 8.17b: FULYP-The residual error, *e(k)* *y-axis: x10^{-7} , x-axis: 4000 iterations*

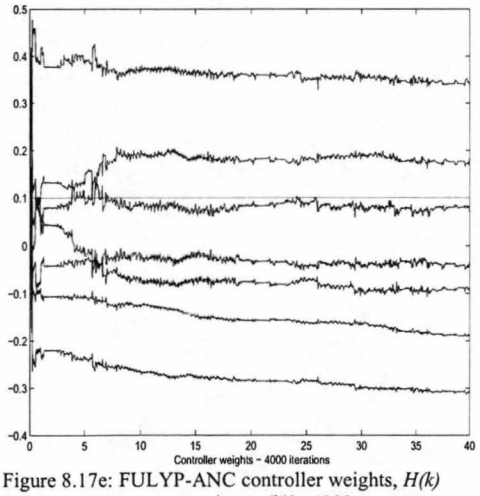
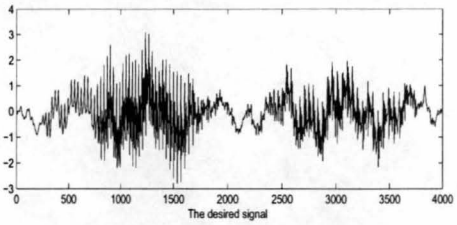


Figure 8.17e: FULYP-ANC controller weights, *H(k)* (without measurement noise, $v_1(k)$), *4000 iterations*



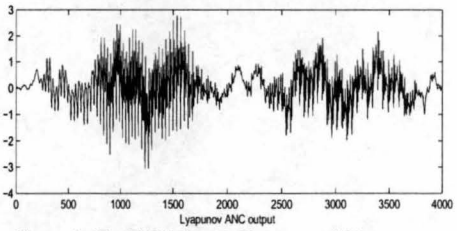


Figure 8.17c: FULMS- ANC output, *y'(k)* & undesired signal, *d(k)*

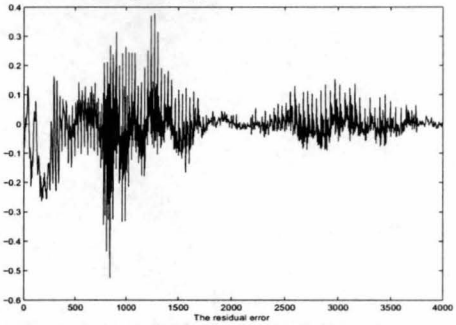


Figure 8.17d: FULMS, the residual error, *e(k)*

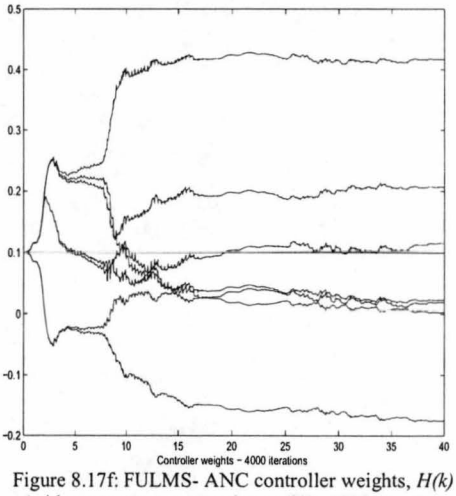


Figure 8.17f: FULMS- ANC controller weights, *H(k)* (without measurement noise, $v_1(k)$), *4000 iterations*

8.5 A Hybrid Nonlinear Neural Predictor And Its Application To Nonlinear And Noisy Time Series Prediction

In this section, we propose a hybrid predictor with Lyapunov theory-based adaptive algorithms. It consists of the following sub-predictors: (1) A *nonlinear sub-predictor* (NSP), which consists of a *multilayer neural network* (MLNN) with a nonlinear hidden layer and a linear output neuron. The algorithm used to update the weights *is Lyapunov stability-based backpropagation* algorithm (LABP) Chapter 6. (2) A *linear sub-predictor* (LSP), which is a conventional *finite-impulse-response* (FIR) filter. Its weights are adaptively adjusted by the LAF algorithm. The NSP that includes nonlinear functions can predict the nonlinearity of the input time series. However the actual time series contains both linear and nonlinear properties, hence the prediction is not complete in some cases. Therefore the NSP prediction error is further compensated for by employing a LSP after the NSP.

In this section, the prediction mechanism and the role of the NSP and LSP are theoretically and experimentally analyzed. The role of the NSP is to predict the nonlinear and some part of the linear property of the time series. The LSP works to predict the NSP prediction error. Lyapunov functions are defined for these prediction errors so that they converge to zero asymptotically. The signals' stochastic properties are not required and the error dynamic stability is guaranteed by the Lyapunov Theory. The design of this hybrid predictor is simplified compared to exiting hybrid or cascade neural predictors [106]-[107]. It is fast convergence and less computation complexity. Furthermore predictability of the hybrid predictor for noisy time series is investigated. The sigmoidal functions used in the NSP can suppress the noise effects by using their saturation regions. Moreover the proposed adaptive algorithms for NSP and LSP are robustness to noisy time series. Computer simulations using nonlinear sunspot times series, real-world data and other conventional predictor models are demonstrated. The theoretical analysis of the predictor mechanism is confirmed through these simulations.

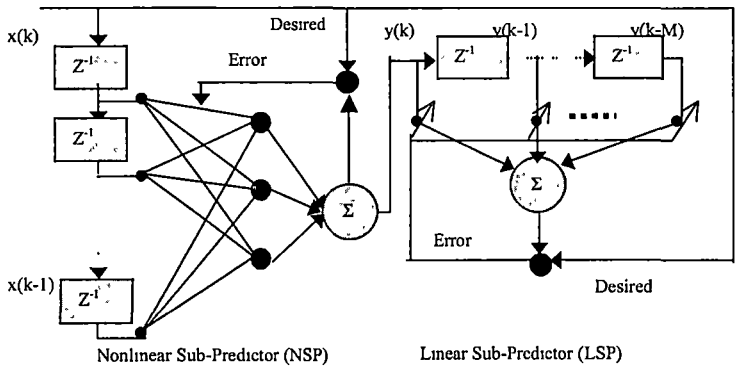


Figure 8.18: Structure of the hybrid predictor

8.5.1 A Hybrid Structure of Neural Network-FIR Predictor

Figure 8.18 illustrates the proposed hybrid predictor structure that is the cascade form of MLNN and FIR filter. The actual time series contains both linear and nonlinear properties and its amplitude is usually continuous value. For these reasons, we combine nonlinear and linear predictors in a cascade form. The nonlinear prediction problem can be described as follow: A set of the past samples *x(k-1),....,x(k-N)* is transformed into the output, which is the prediction of the next coming sample *x(k)*. Therefore we employ a MLNN called the NSP in the first stage. It consists of a sigmiodal hidden layer and a single output neuron. The NSP is trained by the supervised LABP leaning algorithm in Chapter 6. This means the NSP itself acts as a single nonlinear predictor.

In reality it is rather difficult to generate the continuous amplitude and to predict linear property. Hence a linear predictor is employed after the NSP to compensate for the linear relation between the input samples and the target. A FIR filter is used for this purpose, which will be called LSP. The LSP is trained by the LAF. The same target or the desired time series is used for both NSP and the LSP. Hence the nonlinear and some part of linear properties of the input signal can be predicted by the NSP and the remaining part is predicted by the LSP.

8.5.2 Nonlinear Sub-Predictor (NSP)

The architecture of the MLNN considered is shown in *Figure 8.18*. It consists of a hidden layer and a single output neuron. The input $x(k)$ is a sampled signal: $x(k)=\{x(k-1),\ldots x(k-N)\}$ or $x(k)=\{x_k, x_{k-1}, \ldots, x_{k-N}\}$ and the output is a scalar $y_{NSP}(k)$. The purpose of this neural network is to adjust the neural weights in order to achieve error between the network output $y_{NSP}(k)$ and the desired output $d(k)$ converge to zero asymptotically. Let $W_{j,l}^{(l,)}(k)$ denote the connection weight between the i'th neuron in the input layer, $l=0$ and j'th neuron in the hidden layer, $l=1$(for $i = 1, 2, \ldots N; j = 1, 2, \ldots M$). Let $S_j(n)$ and $F_j(\cdot)$ be the output and the activation function of the j'th neuron in the hidden layer, respectively. $W_{1j}^{(2)}(k)$ denotes the connection weight between the j'th neuron in the hidden layer and the neuron in the output layer $y(k)$, $l=2$. Then we have the following system equations:

$$y_{NSP}(k) = \sum_{j=1}^{M} W_{1j}^{(2)} S_j(k) \tag{8.50}$$

$$S_j(k) = F_j\left(\sum_{i=1}^{N} W_{ji}^{(1)} x_i(k)\right) \tag{8.51}$$

where $j = 1, 2, \ldots, M$ and $i = 1, 2, \ldots, N$.

Substituting (8.51) into (8.50) gives

$$y_{NSP}(k) = \sum_{j=1}^{M} W_{1j}^{(2)}(k) F_j\left(\sum_{i=1}^{N} W_{ji}^{(1)} x_i(k)\right) \tag{8.52}$$

where $F(\bullet) = \dfrac{1}{1+e^{-\alpha(\bullet)}}$

The prediction error for NSP is defined as

$$e_{NSP}(k) = d(k) - y_{NSP}(k) \tag{8.53}$$

The learning algorithm for the MLNN can be summarized as:

$$W_{1j}^{(2)}(k) = W_{1j}^{(2)}(k-1) + \Delta W_{1j}^{(2)}(k) \tag{8.54}$$

$$\Delta W_{1j}^{(2)}(k) = \frac{1}{S_j(k)} \frac{1}{M}\left[d(k) - \sum_{j=1}^{M} W_{1j}^{(2)} S_j(k)\right] + \alpha e(k-1) \tag{8.55}$$

and $$W_{ji}^{(1)}(k) = W_{ji}^{(1)}(k-1) + \Delta W_{ji}^{(1)}(k) \tag{8.56}$$

$$\Delta W_{ji}^{(1)}(k) = \left[-W_{ji}^{(1)}(k-1) + \frac{1}{N}\frac{1}{x_i(k)} g_j(u(k)) \right] \tag{8.57}$$

where $$u(k) = \frac{1}{M}\frac{1}{W_{1j}(k)} d(k) \tag{8.58}$$

$$g_j(\bullet) = F_j^{-1}(\bullet) \text{ and } 0 < \alpha < 1 \tag{8.59}$$

or $$\Delta W_{1j}^{(2)}(k) = \frac{1}{S_j(k) + \lambda_1}\frac{1}{M}\left[d(k) - \sum_{j=1}^{M} W_{1j}^{(2)} S_j(k) \right] + \alpha e(k-1) \tag{8.60}$$

$$\Delta W_{ji}^{(1)}(k) = \left[-W_{ji}^{(1)}(k-1) + \frac{1}{N}\frac{1}{x_i(k) + \lambda_2} g_j(u(k)) \right] \tag{8.61}$$

where $$u(k) = \frac{1}{M}\frac{1}{W_{1j}(k) + \lambda_3} d(k) \tag{8.62}$$

The circumstantial derivation and design of the LABP algorithm can be found in Chapter 6.

8.5.3 linear Sub-Predictor (LSP)

The LSP consists a conventional FIR filter. It can be characterized by the difference equation

$$y_{LSP}(k) = \sum_{i=0}^{K-1} h_i(k) y_{NSP}(k-i) \tag{8.63}$$

or $$y_{LSP}(k) = H^T(k) Y_{NSP}(k) \tag{8.64}$$

where $$H(k) = [h_n(0),\ h_n(1),\ ...,\ h_n(N-1)]^T$$

$$Y_{NSP}(k) = [y_{NSP}(k),\ y_{NSP}(k-1),\ ...,\ y_{NSP}(k-N+1)]^T$$

The LSP's coefficient vector is updated by the LAF algorithm in Chapter 3:

$$H(k) = H(k-1) + g(k)\alpha(k) \tag{8.65}$$

$$\alpha(k) = d(k) - H^T(k-1) Y_{NSP}(k) \tag{8.66}$$

$$g(k)=\frac{Y_{NSP}(k)}{\|Y_{NSP}(k)\|^2}\left(1-\kappa\frac{|e_{LSP}(k-1)|}{|\alpha(k)|}\right) \tag{8.67}$$

$$g(k)=\frac{Y_{NSP}(k)}{\lambda_3+\|Y_{NSP}(k)\|^2}\left(1-\kappa\frac{|e_{LSP}(k-1)|}{\lambda_4+|\alpha(k)|}\right) \tag{8.68}$$

where λ_3, λ_4 are small positive numbers and $0 \le \kappa < 1$,

8.5.4 Prediction Analysis

The role of LSP is to predict the prediction error caused by the NSP [108],[109]. It analysis can be summarized as follow:

$$y_{NSP}(k)=d(k)-e_{NSP}(k) \tag{8.69}$$

Due to the LSP is the FIR structure with K taps, its output $y_{LSP}(k)$ can be expressed as

$$y_{LSP}(k)=h_0 y_{NSP}(k)+h_1 y_{NSP}(k-1)+...+h_{K-1}y_{NSP}(k-K+1) \tag{8.70}$$

By substituting the expression (8.69) into (8.70), we get

$$\begin{aligned} y_{LSP}(k) &= h_0(d(k)-e_{NSP}(k))+h_1 y_{NSP}(k-1)+...+h_{K-1}y_{NSP}(k-K+1) \\ &= h_0 d(k)+[-h_0 e_{NSP}(k)+h_1 y_{NSP}(k-1)+...+h_{K-1}y_{NSP}(k-K+1)] \end{aligned} \tag{8.71}$$

Let $$y^*(k)=h_1 y_{NSP}(k-1)+...+h_{K-1}y_{NSP}(k-K+1) \tag{8.72}$$

With the assumption that $h_0 \approx 1$, the expression (8.71) can be rewritten as

$$y_{LSP}(k)=d(k)-[e_{NSP}(k)-y^*(k)] \tag{8.73}$$

Therefore the final prediction error can be expressed as

$$e_{final}(k)=e_{LSP}(k)=d(k)-y_{LSP}(k)=e_{NSP}(k)-y^*(k) \tag{8.74}$$

Hence, the function of LSP is to predict the prediction error resulted from the NSP. The $e_{NSP}(k)$ may include both nonlinearity and linearity. It is noticed that it cannot be predicted by the LSP only if the nonlinearity is dominant. Hence, a NSP is necessitated to predict the nonlinearity. As the result, a hybrid structure is needed. In this hybrid predictor, the prediction mechanism is divided into two stages, the nonlinear and some linear properties of the input time series are prediction by the NSP in the initial stage. In the later stage, the prediction error is further compensated

by the LSP. That is the reason why the same desired response is applied to both LSP and NSP as a target.

The contribution of the NSP and the LSP in the overall performance of the proposed hybrid prediction can be measured by the following ratio

$$R = P_{NSP} / P_{LSP} \tag{8.75}$$

Where P_{NSP} and P_{LSP} are the power of the NSP output and LSP output respectively. The *normalized root mean square error* (NRMSE) [108],[109] is used to express the prediction error so that they can be used for comparison. It is calculated as

$$NRMSE = \sqrt{MSE / P_{input}} \tag{8.76}$$

where MSE indicates the mean squared error of NSP or LSP. P_{input} is the input signal power.

8.5.5 Simulation Results Using Hybrid Model

Example 1: Nonlinear Times Series

Simulations have been done for a one-step ahead prediction of 2 examples*: Sunspot data* and *Chaotic data.* Sunspot data is used as a benchmark for many years by researchers. Data file of the Sunspot times series is download from [110]. It consists the sunspot data from the year 1700 to 1999 (300 Samples). Chaotic time series is used because of its high nonlinearity. *Figures 8.19* and *8.20* show the plots of the sunspot time series and the chaotic time series respectively. *Figure 8.21a* illustrate the plot of the output of the hybrid predictor for sunspot time series (1950-1999). *Figures 8.21b* and *8.21c* show the square predictor error of NSP, $e_{NSP}^2(k)$ and LSP, $e_{LSP}^2(k)$ respectively. *Figures 8.22a, 8.22b, 8.22c* reveal the outputs, y_{LSP} (250-300 samples), $e_{NSP}^2(k)$ and $e_{LSP}^2(k)$ for chaotic time series. Simulation results have shown the hybrid predictor gives good performance.

Example 2: Effects of Noise in Nonlinear Prediction

In measuring physical phenomena, data transmission and processing, noise cannot be neglected. Hence noise effects must be investigated in the real world application. The simulation is carried out using the noisy data as the input and the input and the noise-free data as the desired response. The noise used here is Gaussian white noise. The sigmoidal functions are used in the hidden layer of the MLNN. The noise effects can be suppressed if the noisy input data is distributed mainly in the saturation regions [108],[109]. The LABP algorithm in NSP and LAF algorithm in LSP are robust to additive noise or disturbance even if the noisy data is not distributed in those regions. Hence convergence to zero asymptotically with additive noise can be achieved by this proposed hybrid prediction. *Figure 8.23a* reveals the noisy input data with additive noise. The LSP's square error $e_{LSP}^2(k)$ and NSP's square error, $e_{NSP}^2(k)$ are illustrated in *Figures 8.23b, 8.23c* correspondingly.

Example 3: Comparison With Other Models

In this section, the prediction performance of the proposed hybrid predictor, a linear FIR predictor and a nonlinear MLNN predictor with a linear output neuron are compared for the Sunspot time series. Comparison using different kinds of predictor was demonstrated in [108]. The simulation results using the Sunspot time series are tabulated in *Table 8.4.* The MLNN predictor is trained with BP and same predictor size or parameters are with the MLNN of NSP in the proposed hybrid predictor. The linear predictor used in the simulation is FIR filter trained with LMS to examine the efficiency using LSP only. Compared to those models, the proposed hybrid predictor has the minimum prediction errors in both cases. The linear predictor does not perform well due to the high nonlinearity in the time series.

Conclusively, a hybrid nonlinear time series predictor that consists the NSP and the LSP combined in a cascade form is proposed. Simulations have been demonstrated using the linear FIR with LMS predictor, nonlinear MLNN with BP predictor and the hybrid predictor with combination formal MLNN and linear FIR for comparison. Properties of these predictors are analyzed taking the nonlinearity of the time series

into account. Hence the prediction mechanism and the role of the NSP and LSP of the hybrid predictor have been theoretically and experimentally analyzed and clarified.

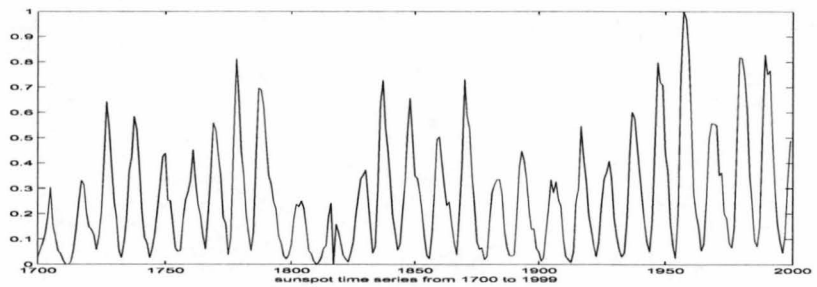


Figure 8.19: Sunspot Time Series from 1700 to 1999

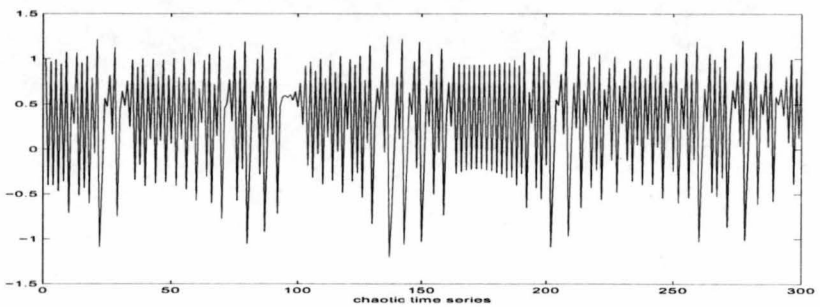


Figure 8.20: Chaotic Time Series used in the simulation

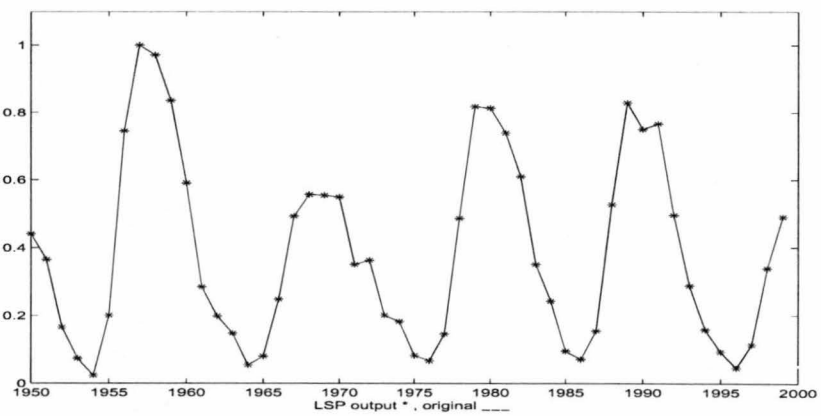


Figure 8.21a: Predictor output waveforms for Sunspot data using the proposed hybrid predictor (from 1950-1999) '___ original data', ' * predictor output data'

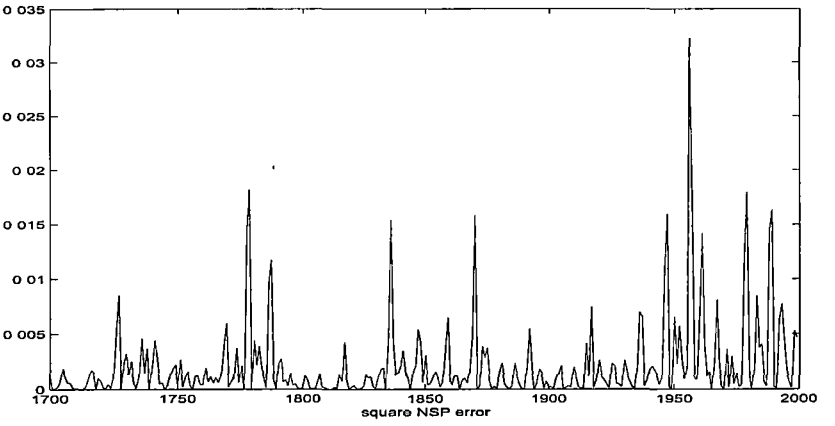


Figure 8.21b: The NSP square output error, $e_{NSP}^2(k)$

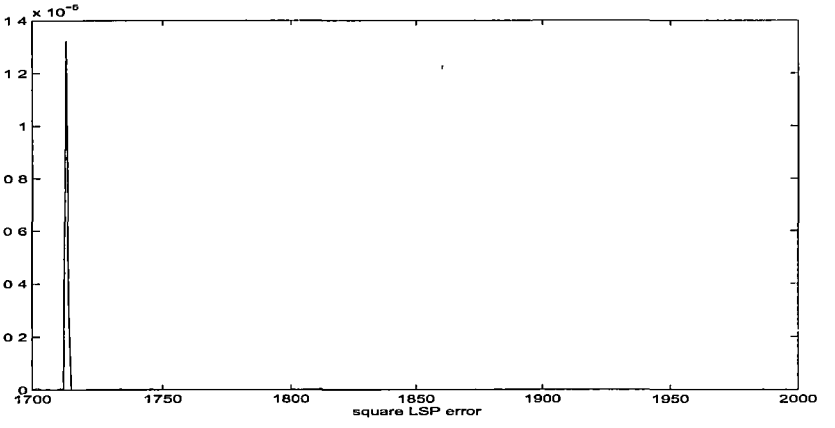


Figure 8.21c: The LSP square output error, $e_{LSP}^2(k)$

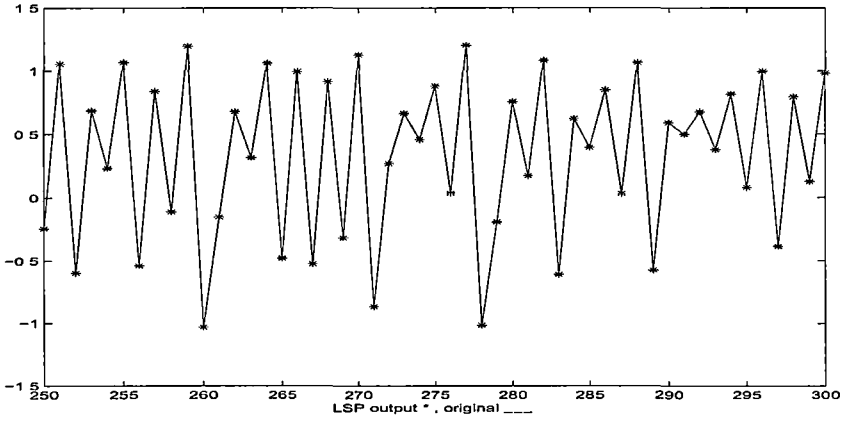


Figure 8.22a: Predictor output waveforms for Chaotic data using the proposed hybrid predictor (250-300 samples) '___ original data', ' * predictor output data'

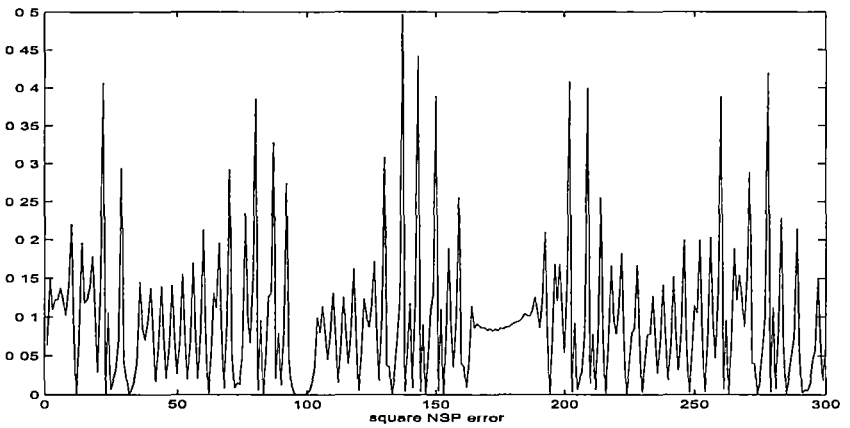


Figure 8.22b: The NSP square output error, $e_{NSP}^2(k)$

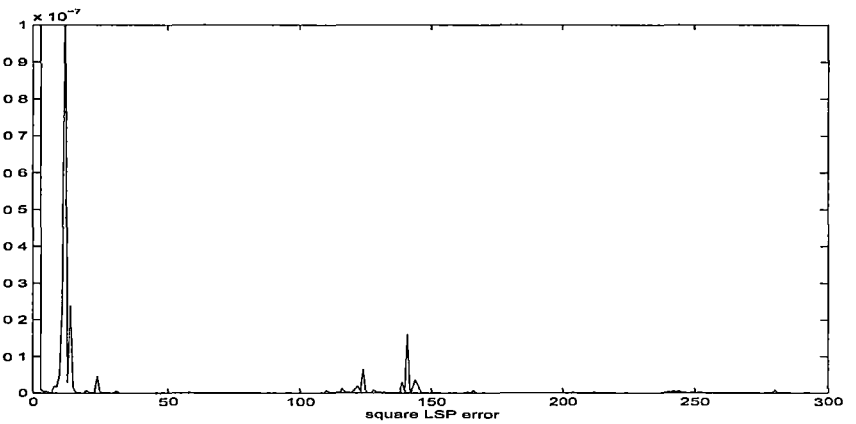


Figure 8.22b: The LSP square output error, $e_{NSP}^2(k)$

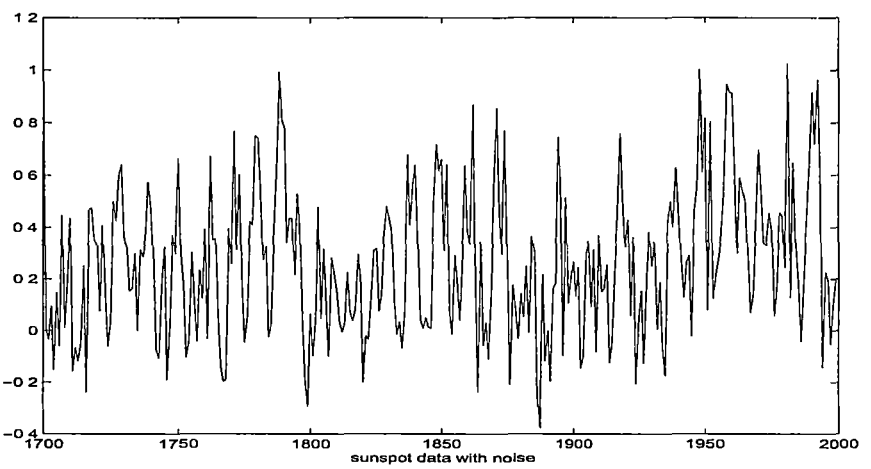


Figure 8.23a: Sunspot Time Series from 1700 to 1999 + Gaussion noise

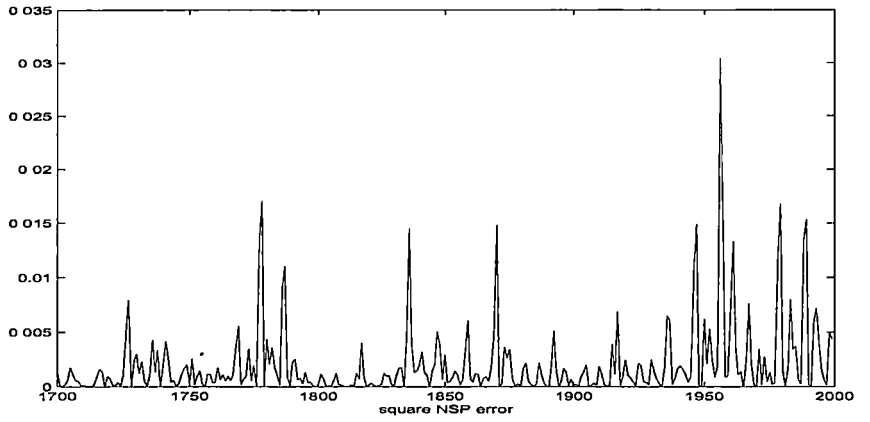


Figure 8.23b: The NSP square output error, $e_{NSP}^2(k)$

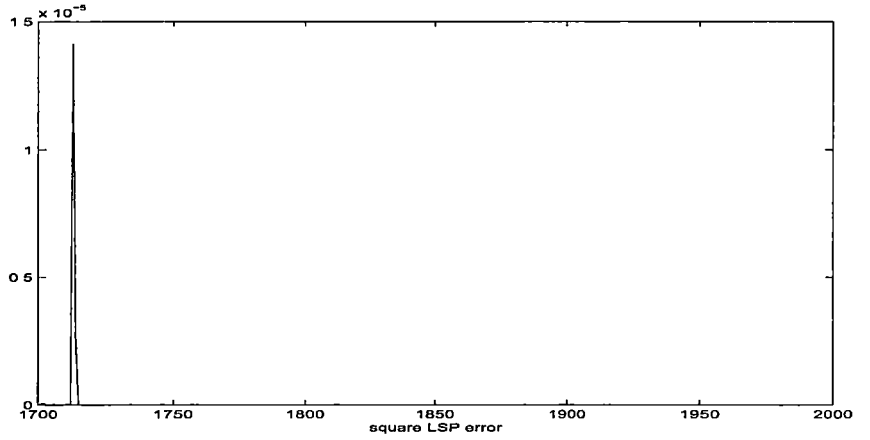


Figure 8.23c: The LSP square output error, $e_{NSP}^2(k)$

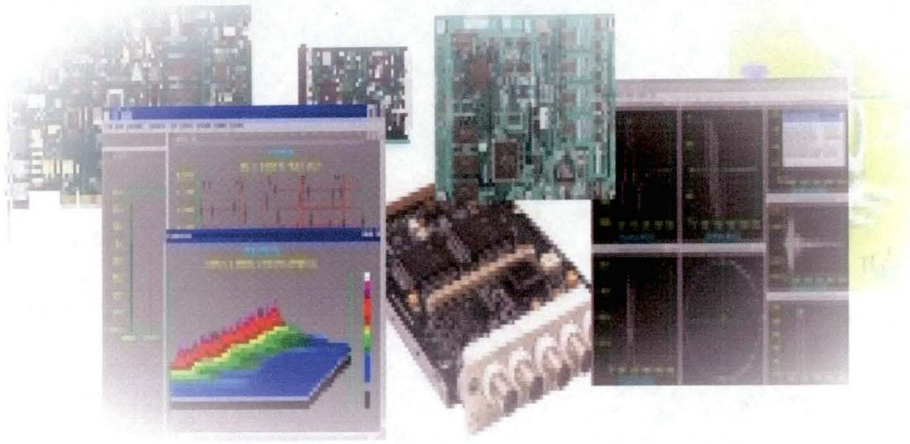
Table 8.4: Comparison of NRMSE among different models for sunspot data

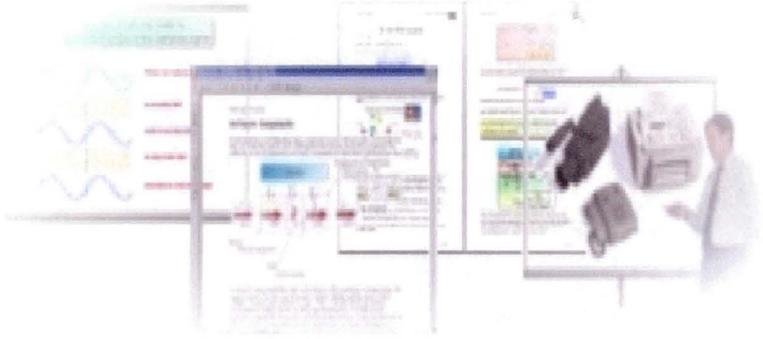
Model	**Proposed Hybrid Predictor**	**MLNN Predictor (BP)**	**Linear FIR predictor (LMS)**
NRMSE	4.6×10^{-4}(NRMSE of LSP)	0.092	0.2897
	0.091 (NRMSE of NSP)		

8.6 Conclusion

This chapter has presented various adaptive filtering schemes that are modified based on the adaptive filtering techniques presented in Chapter 3-6. It has also explored the specific applications for these schemes. The theoretical analysis and simulation results have indicated the proposed methods can offer good performance. Hopefully, further researches to modify the proposed schemes in Chapter 3-7 can be carried out so that these modified methods can be applied to different applications.

Chapter 9





Conclusions

Chapter 9

Conclusions

9.1 Contributions and Summary

This thesis has provided a fundamental breakthrough in understanding of the Lyapunov stability-based adaptive filtering mechanism, yielding further conditions and solutions for a number of nonlinear filtering problems using Lyapunov stability theory . Further investigations include the theory and design of RBF neural network-based nonlinear adaptive filters with Lyapunov stability, fuzzy adaptive filters with Lyapunov sense fuzzy rules, neural adaptive filters with the back-propagation learning rules in Lyapunov sense with guaranteed stability, polynomial adaptive filters with Lyapunov stability and parallel signal processing using Lyapunov theory. These new adaptive filtering schemes have been tailored to different applications. Simulation examples have been performed to investigate various performances such as tracking precision, stability, and robustness of the developed schemes. The successful outcome of the thesis will in no doubt make significant contributions to and impacts on research in the field of intelligent signal processing and communications systems. Applications and commercial potential of this research in signal processing, telecommunications (both wired and wireless) and many other industries cannot be underestimated.

The contributions or benefits of this thesis to signal processing theory and applications currently include the asymptotic error convergence as a new paradigm in the area of adaptive signal processing. This will greatly improve the transient performance between the reference signal and the output of the adaptive filters. Furthermore, the constructive procedures for adaptive filter designs that integrate this

new mechanism will permit high precision for signal processing. Unlike the design of conventional adaptive filters, the performance index in the weight space is not used. A performance index in the error space with a single global minimum is created based on the Lyapunov stability theory. Therefore, local minima encountered in traditional methods are avoided in the search of the optimal filter parameters. In addition, the design of adaptive filters in this thesis is independent of the stochastic properties of the signals because only the desired reference signal and the observations of the filter output are used for the design of the adaptive filters. Therefore, the algorithms to be developed in this thesis are suitable for both random processes and deterministic processes. In summary, this thesis has been mainly concerned with the study and improvements of adaptive filtering schemes by employing Lyapunov theory and artificial intelligent technologies.

Chapter 2 of this thesis has provided a brief survey of adaptive filtering theory. It constitutes mostly review material. It provides a background on existing techniques for adaptive filtering. Introduction, the performance measures, system configurations of adaptive filter are presented. Adaptive filter models such as *finite impulse response* (FIR) and *infinite impulse response* (IIR) are discussed. Adaptive algorithms for FIR and IIR filters are then reviewed. This chapter has mainly concerned the linear adaptive filtering.

Chapter 3 has contributed the mathematical foundation of adaptive filter designs using Lyapunov stability theory. A new adaptive filtering technique called *Lyapunov Theory-based Adaptive Filtering* (LAF) has been developed. A Lyapunov function of the error between the desired signal and the filter output is defined, the weights of the filter are then adaptively adjusted based on Lyapunov stability theory so that the error can asymptotically converge to zero. For conventional adaptive filter designs, a performance index of the error is defined first and the performance index is then minimized in the weight or parameter space. If the performance index is complex with many local minima, the optimization may stop at some local minimum for some initial values of the filter parameters. In such a situation, the optimal filter parameters cannot be obtained. In contrast, using the LAF design approach, the above difficult can be avoided. In this new approach, the update law of the adaptive filter is not used to search the global minimum of the performance index in the weight or parameter

space. Instead, the update law of the adaptive filter is designed where the performance index will have only one global minimum in the error space when the time tends to infinity. Because the selected Lyapunov function, *V(k)* is positive definite and the update law is designed such that the difference of *V(k+1)* and *V(k)* is negative, the value of the performance index *V(k)* will therefore tend to zero, which is the global minimum of the performance in the error space. Compared with the traditional optimization theory [1]-[3], Lyapunov stability theory can be treated as an optimization method in the error space. Therefore, investigation of the optimization problem using Lyapunov stability theory in the error domain for adaptive IIR filter with local minima are presented, which are not achievable using traditional optimization approaches. Although the input signal of the adaptive filter is disturbed by the bounded random noises, only the input and the output measurements are used for the design of the Lyapunov filters. Therefore, the design of Lyapunov adaptive filters is independent of the stochastic properties of the random input disturbances. Further investigations that explore the convergence rate of the error of the LAF filters have also been developed. The convergence region for the error of the modified Lyapunov filter in order to avoid the singularities has been discussed.

In chapter 4, the design of adaptive filters using *radial basis function* (RBF) neural networks and Lyapunov stability theory is presented. It is well known in the area of artificial neural networks that an RBF neural network, which consists of Gaussian type of nonlinear function nodes, a linear input layer, a nonlinear hidden layer and a linear output layer, has the ability to approximate arbitrary linear or nonlinear mapping through learning. In this chapter, two realizations of the Lyapunov adaptive filters using RBF neural networks are proposed. The FIR and IIR filters are configured as feedforward and recurrent RBF networks respectively. Unlike many adaptive neural filtering schemes using gradient search in the parameter space, the selected Lyapunov function for the adaptive RBF filter has a unique global minimum in the state space. By properly choosing the weights update law in Lyapunov sense, the output of the adaptive RBF neural filter can asymptotically converge to the desired reference signal. Thus the local minima problem occurred in the gradient search-based adaptive filters is avoided,. Although the input signal of the RBF neural filter is disturbed by the bounded random noises, only the input and the output

measurements are needed for the design of the RBF neural filters. Hence the proposed scheme is independent of the statistical properties of the input signals.

Chapter 5 has provided the design of fuzzy adaptive filters using Lyapunov theory. Fuzzy adaptive filtering has received a great deal of attention recently. Fuzzy rules are devised by human experts, based on observations and the measurement data, in order to adjust the filter parameters. Then fuzzy logic technology (fuzzification, reasoning, and defuzzification) is used to obtain the output of the fuzzy filter. Similar to fuzzy control, the tracking performance, robustness, and the stability of the fuzzy filters are difficult to analyze. In this chapter, two types of adaptive fuzzy filters are developed using fuzzy logic and Lyapunov stability theory to design new adaptive filters to overcome the disadvantages of the previous fuzzy adaptive filters. First, a *fuzzy gain Lyapunov adaptive filter* for nonlinear adaptive filtering has been proposed. It incorporates fuzzy logic to the LAF by the use of a set of Lyapunov sense fuzzy if-then rules. Given the input signal and its squared norm, these rules are then used to determine the adaptive gain to update the filter parameters. The second fuzzy adaptive filter is named *LAF fuzzy adaptive filter*. This fuzzy adaptive filter is constructed from a set of changeable fuzzy IF-THEN rules. The LAF is used to update the parameter of the membership functions so that the dynamic error between the filter output and the desired response converges to zero asymptotically. Therefore, the most significant advantage of the fuzzy filter compared to the conventional filters is that linguistic information from human experts (in the form of fuzzy IF-THEN rules) can be incorporated into the filter.

In the chapter 6, we have developed the design of neural adaptive filters with the backpropagation (BP) learning rules in Lyapunov sense. The existing BP learning rules obtained from the optimization theory have been widely used for neural network-based adaptive filter designs [26],[53]. As mentioned in [26],[53], the BP learning rules are used to search the optimal parameters of the neural adaptive filters in the weight space. Because of the nonlinearity of neural networks, and the complexity of the performance index, the analysis on the tracking precision, stability, and the robustness of the neural adaptive filters cannot be carried out. In this chapter, Lyapunov stability theory has been used, instead of the traditional optimization methods, to update the weights of the neural adaptive filters. New BP learning rules

will be developed to improve the error precision and the stability of the closed loop adaptive filter systems. We call these new learning rules as *Lyapunov Stability-based Adaptive Backpropagation* (LABP) algorithm. It is expected that the LABP learning rules will significantly improve the performance of adaptive filters and can also be extended other areas.

Chapter 7 has presented one area of nonlinear signal processing known as polynomial signal processing using Lyapunov theory. The first part of this chapter presents a fast, less computation complexity and stable adaptive polynomial filters. We only focus on the following polynomial models: (1) *Volterra model* that the nonlinear system output signal can be related to the input signal through a truncated Volterra series expansion. (2) *Bilinear model* that involves and recursive nonlinear difference equation. The second part of the chapter considers another realization of nonlinear Volterra filter using *parallel-cascade* structure. Parallel-cascade realizations implement higher order Volterra systems as a parallel connection of multiplicative combinations of lower order truncated Volterra systems. All the proposed techniques in this chapter have excellent convergence and their stability are guaranteed by the Lyapunov stability theory. These schemes are independent of signals' stochastic properties. They have less or comparable computational complexity compared to some conventional polynomial filters.

Chapter 8 has introduced other different techniques based on those schemes proposed in Chapter 3-Chapter 6. These techniques include (1) A new concurrent algorithm for adaptive filtering called *concurrent Lyapunov theory-based adaptive filtering* (CLAF) in parallel signal processing. (2) Complex-valued Lyapunov theory-based adaptive filtering (Complex-LAF). (3) A new approach in feedforward active noise control using Lyapunov stability theory which consists two algorithms called *Filtered-X Lyapunov theory-based* algorithm (FXLYP), *Filtered-U Lyapunov theory-based* algorithm (FULYP), and a overall on-line modeling techniques using the LAF. (4) A hybrid nonlinear neural predictor and its application to nonlinear and noisy time series prediction. Most of these methods are the modification of the schemes presented in Chapter 3-7 so that they can be applied to particular applications.

9.2 Further Extensions And Developments

This thesis has touched a number of different areas of research including adaptive filtering, neural networks, fuzzy logic, polynomial signal processing, parallel signal processing and different applications. In each of these areas, various techniques have been developed using Lyapunov theory and artificial intelligent techniques. Despite the good achievements obtained from some of these approaches, there is inevitably room for improvements and extensions within of relating to the scope of this study. Some areas for further research related to the thesis are:

1. *Different Lyapunov functions and weight or filter parameter updated laws.* The further research-based that can be carried out is to use different Lyapunov functions and different parameter updated laws or weight learning rules to further improve the convergence properties and the robustness properties of the different schemes or algorithms proposed in Chapter 3-8 with respect to the bounded random disturbances or other aspects.

2. *Further Theoretical and Experimental Work on the LABP in Chapter 6 for the MLP with more than one hidden layer.* It is well-known that multilayer perceptron (MLP) with one hidden layer and sufficient hidden units can approximate any arbitrary nonlinear function. Although MLP with two hidden layers or more layers may give better approximation for some specific problem, MLP with two hidden layers or more prone to fall into the local minima. The proposed LABP can provide a solution to the problem. Therefore the future works on the LABP for MLP with two or more hidden layers are necessary.

3. *Further Research on the LAF Fuzzy Adaptive Filter in Chapter 5 and Realization of Nonlinear Volterra filter using Parallel-Cascade Structure in Chapter 7.* The results presented on LAF fuzzy adaptive filter and nonlinear filtering (parallel-cascade structure) are preliminary results. Further theoretical and experimental works are needed to performance to further support the proposed methods.

4. *The Real-time Implementation of the New Adaptive Filters for Audio Signal Processing.* The new adaptive filtering methods can be laboratory tested in the

future hopefully. For this purpose, an audio signal processing system can be designed henceforth. In this audio signal processing system, an advanced microprocessor will be used, and the adaptive filters for narrow band noise cancellation can be implemented readily to improve the quality of audio signals. Major tasks include interfacing the adaptive filters with the computer and the audio signals. Major tasks include interfacing the adaptive filters with the computer and the audio signal processing system, computer programming, and the adjustment of the signal processing software and hardware. The performance of this new audio system will be compared with that of an audio signal processing system with a traditional adaptive filter using LMS algorithm implemented on the Texas Instrument; TMS320C30 EVM. Criteria to be used include stability, convergence rate, tracking precision as well as general audio signal quality.

5. *The Real-time Implementation of the New Adaptive Filters for Image Processing.* Another important application area is image processing. Multi-dimensional adaptive filtering algorithms will be developed based on one-dimensional adaptive filtering schemes. The new adaptive filtering methods will be real- time implemented used for image processing in the future. The performance of the new multi-dimensional adaptive filtering techniques will be compared with that of the existing high performance methods such as Wiener filtering and POCS (Projection Onto Convex Sets) in post-filtering applications to eliminate or reduce digital image and video coding distortions henceforward.

References

[1] S. Haykin, Adaptive filter theory. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[2] C.F.N. Cowan, P.M. Grant, Adaptive Filters: Prentice-Hall Signal Processing Series, 1985

[3] Bernard Mulgrew, Colin F. N. Cowan, Adaptive filters and equalizers: Kluwer Academic Publishers, 1984.

[4] G. Carayannis, D. Manolakis, N. Kalouptsidis, "A fast sequential algorithm for least-squares filtering and prediction", IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-31, Dec. 1983.

[5] M. L. Honig, D. G. Messerschmitt. Adaptive filters, structures, algorithms, and applications. Boston, MA:Kluwer.

[6] J. G. Proakis, D. G. Manolakis. Introduction to digital signal processing. New York: Macmillan.

[7] Dirk T. M. Slock, Thomas Kailath, "Numerical stable fast transversal filters for recursive least squares adaptive filtering", IEEE Trans. Signal Processing, vol. 39, no. 1, 1991.

[8] J. M. Cioffi, T. Kailath, "Fast, recursive least squares transversal filters for adaptive filtering", IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-32, no. 2, pp. 304-337, April, 1984.

[9] Mariusz Zoltowski, "Why do optimal forgetting RLSs exhibit long term divergence and how can this be avoided?", Signal Processing, vol. 68, pp. 195-218, 1998.

[10] J. M. Cioffi, T. Kailath, "Windowed fast transversal filters adaptive algorithms with normalization", IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-33, no. 3, pp. 607-625, June 1985.

[11] M. Mueller, "Least-squares algorithms for adaptive equalizers", Bell Syst. Tech. J., vol. 60, pp.1905-1925, 1981.

[12] S. Ljung and L. Ljung, "Error propagation properties of least squares adaptation algorithms", Automatica, vol. 21, pp. 157-167, 1985.

[13] D. Falconer and L. Ljung, "Application of fast Kalman estimation to adaptive equalization", IEEE Trans. Commum., vol. COMM-26, pp. 1439-1446, 1978.

[14] M. Verhaegen, "Roundoff error propagation in four generally applicable, recursive least squares estimation schemes", Automatica, vol. 25, no. 3, pp.437-444, 1989.

[15] Mangesh M. Chansarkar and Uday B. Desai, "A robust least squares algorithm", IEEE Trans. on Signal Processing, Vol. 45, no. 7, 1997.

[16] L. Ljung, "Analysis of recursive stochastic algorithms", IEEE Trans. Automat. Contr., Vol. AC-22, pp. 551-575, 1977.

[17] M. Verhaegen, "Roundoff error propagation in four generally applicable, recursive least squares estimation schemes", Automatica, vol. 25, no. 3, pp.437-444, 1989.

[18] J. J. Shynk, "Adaptive IIR Filtering" *IEEE ASSP Magazine,* pp. 4-12, November 1989.

[19] Man ZhiHong, H.R.Wu, W.Lai and Thong Nguyen, "Design of Adaptive Filters Using Lyapunov Stability Theory", The 6th IEEE International Workshop on Intelligent Signal Processing and Communication Systems, vol, pp. 304-308, 1998.

[20] Slotine, J-J. E. and Li, W. Applied nonlinear control, Prentice-Hall, Englewood Cliffs, NJ, 1991

[21] J J Shynk, "Adaptive IIR Filtering Using Parallel-Form Realizations", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 37, no. 4, pp.519-531, April 1989.

[22] X Y Gao and W M Snelgrove, "An Efficient Adaptive Cascade IIR Filter", IEEE Transactions on Circuits and Systems II, vol. 39, no. 9, pp. 444-447, September 1992.

[23] Phillip A. Regalia, Adaptive IIR filtering in signal processing and control. Marcel Dekker.

[24] D. Parikh, N. Ahmed, and S. D. Stearns, "An adaptive lattice algorithm for recursive filters," IEEE Trans. Acoust., Speech, Sig. Proc., vol. ASSP-28, no.1, pp. 110-111, Feb. 1980.

[25] L. Ljung, and T. Söderström, Theory and Practice of Recursive Identification. MIT Press, Cambridge, 1983.

[26] Fa-Long Luo, Rolf Unbehauen, Applied neural networks for signal processing: Cambridge University Press, 1997.

[27] Nerrand O., Roussel-Ragot P., Personnaz L., Dreyfus G., Marcos S., "Neural Networks & Nonlinear Adaptive Filtering: Unifying Concepts & New Algorithms", Neural Computation 5: 165-199, 1993.

[28] Stephen W. Piché, "Steepest Descent Algorithms for Neural Network Controllers and Filters", IEEE Trans. Neural Network, vol. 5, no. 2, pp. 198-212, 1994.

[29] Pushkorius, G.V., & Feldkamp, L.A., "Decoupled extended Kalman training of feedfoward layered networks", In Proc. Of the Inter. Joint Conf. On Neural Networks, I, Seattle, pp. 771-777, 1991.

[30] Steve A. Billings, Chi F. Fung, "Recurrent Radial Basis Function Networks for Adaptive Noise Cancellation", Neural Networks, vol.8, no.3, pp. 273-290, 1995.

[31] Monica Bianchini, Marco Gori, "On the problem of local minima in recurrent neural networks", Trans. Neural Network, vol. 5, no. 2, pp. 167-177, 1994.

[32] Monica Bianchini, Paolo Frasconi, Marco Gori, "Learning without Local Minima in Radial Basis Function Networks", Trans. Neural Network, vol. 6, no. 3, pp. 749-755, 1995.

[33] D.E. Goldberg, Genetic Algorithms in Search, Optimization, Machine Learning. Reading, MA: Addison-Wesley, 1989.

[34] K. S. Narenda and M. A. L. Thathachar, Learning Automata - An Introduction. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[35] S. Kirkpatrick, C. Gelatt, M. Vecchi, "Optimization by simulated annealing", Science, vol. 220, pp. 671-680, 1983.

[36] Chen S., Bernard Mulgrew, Peter M. G., "A Clustering Technique for Digital Communications Channel Equalization Using Radial Basis Function Networks", IEEE Trans. Neural Networks, vol. 4, no. 4,pp. 570-579, 1993.

[37] Chi F. Fung, Steve A. B, Wan Luo, "On-line Supervised Adaptive Training Using Radial Basis Function Networks", Neural Networks, vol. 9, no. 9, pp. 1597-1617, 1996.

[38] Pearlmutter B.A., "Gradient Calculations for Dynamic Recurrent Neural Networks: a Survey", IEEE Trans. Neural Networks, vol. 6, no. 5, 1995.

[39] Williams, R. J & Peng, J, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories", Neural Computation, 2, 790-501.

[40] Werbos P.J., "Backpropagation through time: what it does and how to do it", Proc. of IEEE, Special issue on neural networks, vol. 78, no. 10, pp. 1550-1560, 1990.

[41] Williams R.J., Zipse D., "Gradient-based learning algorithms for recurrent networks and their computational complexity", in Backpropagation: Theory, Architectures and Applications, Y. Chauvin and D.E. Rumelhart, Eds. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 433-486, 1994.

[42] Campolucci P., Uncini A., Piazza F., Rao B.D, "On-line learning algorithms for locally recurrent neural networks", IEEE Trans. Neural Networks, vol. 10, no. 2, March 1999.

[43] Tsoi A.C, Back A.D., "Locally recurrent globally feedforward networks: a critical review of architectures ", IEEE Trans. Neural Networks, vol. 5, no. 2, pp. 229-239, March 1994.

[44] Back A.D., Tsoi A.C, "FIR and IIR synapses, a new neural network architecture for time series modelling", Neural Computation 3:375-385, 1991.

[45] Li-Xin Wang and Jerry M. Mende, "Fuzzy Adaptive Filters, with Application to Nonlinear Channel Equalization", IEEE Trans. on Fuzzy System, Vol 1, No. 3, pp. 161-169, 1993.

[46] Li-Xin Wang and Jerry M. Mendel, "Fuzzy Basis Functions, Universal Approximation, and Orthogonal Least Squares Learning,", IEEE Trans. on Fuzzy System, Vol. 3, No 5, pp. 807-814, 1992.

[47] K. Y. Lee, "A Fuzzy Adaptive Decision Feedback Equalizer", IEE Electron. Lett., Vol. 30, No. 10, pp. 749-751, 1994.

[48] Von Altrock, C. Fuzzy Logic and Neurofuzzy Applications Explained, Prentice-Hall, 1995.

[49] Ki Yong Lee, "Complex Fuzzy Adaptive Filter with LMS Algorithm", IEEE Trans. on Signal Processing, Vol. 44, No. 2, Feb 1996.

[50] Ki Yong Lee, "Complex RLS Fuzzy Adaptive Filter And Its Application To Channel Equalization", Electronics Letters, Vol. 30, No. 19, Sept 1994.

[51] S. K. Phooi, Man Zhihong, H.R. Wu, "Nonlinear adaptive RBF neural filter with Lyapunov adaptation algorithm and its application to nonlinear channel equalization", Proc. of the Fifth Int. Symposium on Signal processing and its applications, Brisbane, August, vol. 1 pp. 151-154, 1999.

[52] Chen S, C.F.N. Cowan, P.M.Grant, "Orthogonal least squares learning algorithm for radial basis function networks", IEEE Trans on Neural Networks, vol. 2, no.2, pp. 302-309, 1991.

[53] S. Haykin, Neural Network: A comprehensive foundation. New York: IEEE Press, 1994.

[54] E. A. Wan, "Temporal backpropagation for FIR neural networks", Proc. Int. Joint Neural Network, vol. 1 pp. 577-580, 1990.

[55] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano and K. J. Lang, "Phoneme recognition using time-delay neural network", IEEE Trans. Acoust., Speech, Signal Processing, vol. 37, Mar, 1989.

[56] S. P. Day and M. R. Davenport, "Continuous-time temporal backpropagation with adaptable time delay neural network", Neural Network, vol. 8, no. 3, pp. 447-461, 1995.

[57] F. Beaufays, E. Wan, "Relating real-time backpropagation and backpropagation through time: An application of flow graph interreciprocity ", Neural Comput., vol. 6, pp. 296-306, 1994.

[58] K. S. Narendra, K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing nerual networks", IEEE Trans. Neural Network, vol. 2, Mar, 1991.

[59] J. L. Elman, "Finding structure in time", Cognitive Sci., vol. 14, pp. 179-211, 1990.

[60] P. Frasconi, M. Gori, G. Soda, "Local feedback multilayered networks", Neural Comput., vol. 4 pp. 120-130, 1992.

[61] F. Pineda, "Generalization of backpropagation to recurrent neural networks", Phys. Rev. Letter, vol. 59, no. 19, pp. 2229-2232, Nov. 9, 1987.

[62] L. B. Almeida, "A learning rule for asynchronous perceptrons with feedback in combinatorial environment", Proc. Int. Neural Networks, vol. 2, pp. 609-618, 1987.

[63] B. Srinivasan, U. R. Prasad, N. J. Rao, "Backpropagation through adjoints for the identification of nonlinear dynamic systems using recurrent neural models", IEEE Trans. Neural Network, pp. 213-228, Mar, 1994.

[64] C. Ku, K. Y. Lee, "Diagonal recurrent neural networks for dynamic systems control", IEEE Trans. Neural Network, vol. 6, Jan, 1995.

[65] R. J. Williams and D. Zipser, "Experimental analysis of the real-time learning algorithm ", Cognitive Sci., vol. 1, no. 1, pp. 87-111, 1989.

[66] K. S. Narendra, K. Parthasarathy, "Identification and control of dynamics systems using neural networks", IEEE Trans. Neural Network, no.1, pp. 4-27, 1990.

[67] Zhihong Man, Seng Kah Phooi, H.R. Wu, "Lyapunov stability-based adaptive backpropagation for discrete and continuous time systems", 2nd International Conference on Information, Communications and signal processing, ICICS'99, pp. paper no. 376, 1999.

[68] P. Fransconi, M. Gori, A Tesi, "A Technical note on the convergence of backpropagation to optimal solution", Technical report DSI-TR-1/92 Universita' di Firenze, 1992.

[69] Battiti, R, "First and second order methods for learning: between steepest descent and Newton's method", Neural Computation, vol 4, pp. 141-166, 1992.

[70] G.W.Ng, Application of Neural Networks to Adaptive Control of Nonlinear Systems, Control Systems Centre Series, John Wiley & Son, 1997.

[71] Kollias, S., & Anastrassiou, D., "An adaptive least square algorithm for the efficient training of artificial neural networks", IEEE Trans. Circuit & Systems, vol. 36, pp. 1092-1101, 1989.

[72] Singhal, S., Wu, L., "Training feedback networks with the extended Kalman algorithm", Proc. Of the Inter. Conf. on Acoustics, speech and signal processing, scotland, pp. 1187-1190, 1989.

[73] Webb, A.R., Lowe, D., Bedworth, M.D., "A comparison of nonlinear optimization strategies for feedforward adaptive layered networks", Royal signals & Radar establishment, Memorandum, no. 4157, 1988.

[74] V. John Mathews, "Adaptive Polynomial Filters", IEEE SP Magazine, pp. 10-26, July 1991.

[75] R. W. Brockett, "Volterra series and geometric control theory", Automatica, vol. 12, pp. 167-176, 1976.

[76] R. R. Mohler, W. J. Kolodziej, "An overview of bilinear system theory and applications", IEEE Trans. Syst., Man, Cybern., vol. SMC-10, pp. 683-688, Oct, 1980.

[77] X. Y. Gao, W. M. Snelgrove, D. A. Johns, "Nonlinear IIR adaptive filtering using a bilinear structure", Proc. IEEE Int. Symp. Circuits Syst., Portland, OR, May 1989.

[78] X. Yang, R. R. Mohler, R. M. Burton, "Adaptive suboptimal filtering of bilinear systems", Int. J. Control, vol. 52, no. 1, pp. 135-158, 1980.

[79] M. Korenberg, "Identifying nonlinear difference equation and functional expansion representations: the fast orthogonal algorithm", Ann Biomed. Eng., vol. 16, pp. 132-142, 1988.

[80] Junghsi Lee, V. John Mathews, "A fast recursive least squares adaptive second order volterra filter and its performance analysis", IEEE Signal Processing, vol. 41, no. 3, pp. 1087-1102, 1993.

[81] Thomas M. Panicker, V. John Mathews, "Parallel-Cascade Realizations and Approximations of Truncated Volterra Systems", IEEE Trans. Signal Processing, vol. 46, no. 10, pp. 2829-2832, Oct 1998.

[82] Thomas M. Panicker, V. John Mathews, "Adaptive Parallel-Cascade Truncated Volterra Filters", IEEE Trans. Signal Processing, vol. 46, no. 10,pp. 2664-2673, Oct 1998.

[83] Y. Lou, C. L. Nikias, A.N. Venetsanopoulos, "Efficient VLSI array processing structures for adaptive quadratic digital filters", Circuit Syst. Signal Process., vol. 7, no. 2, pp. 253-273, 1988.

[84] S. Marsi, G.L. Sicuranza, "On reduced-complexity approximation of quadratic filters", Pro. 27th Asilomar Conf. Signals, Syst., Comput., Pacific Grove, CA, pp. 1026-1030, Nov 1993.

[85] Ajit Kumar Chaturvedi, Govind Sharma, "Anew Family of Concurrent Algorithms for Adaptive Volterra and Linear Filters", IEEE Trans. on Signal Processing, Vol. 47, no. 9, 1999.

[86] Chen, S. S. McLaughlin, B. Mulgrew,"Complex-value radial basis function network, Part I: Network architecture and learning algorithm", Signal Processing, Vol. 35, pp. 19-31, 1994.

[87] Chen, S. S. McLaughlin, B. Mulgrew,"Complex-value radial basis function network, Part II: Application to digital communications channel equalization", Signal Processing, Vol. 36, pp.175-178, 1994.

[88] S. J. Elliott, P.A. Nelson, "Active noise control", IEEE Signal Processing Mag., vol. 10, pp. 12-35, Oct., 1993.

[89] H. K. Pelton, S. Wise, W.S. Sims, "Active HVAC noise control systems provide acoustical comfort", Sound and Vibration, pp. 14-18, July 1994.

[90] B. Sanito, "Electronic automobile muffler quiets the skeptics", EE Times, October, 1992.

[91] S. M. Kuo and D. R. Morgan, Active Noise Control Systems: Algorithms and DSP Implementations. New York, Wiley, 1996.

[92] P.A. Nelson, S.J. Elliott, Active Control of Sound. Academic Press INC., 3rd ed., 1995.

[93] C. R. Fuller, A. H. von Flotow, "Active control of sound and vibration", IEEE Control Syst, Mag., vol. 15, pp. 9-19, Dec. 1995.

[94] E. F. Berkman, E. K.. Bender, "Perspectives on active noise and vibration control", Sound Vibration, vol. 31, pp. 80-94, Jan. 1997.

[95] M. O. Tokhi, R. R. Leitch, Active noise control. Oxford, U.K: Clarendon, 1992.

[96] P. Leug, "Process of silencing sound oscillations", U.S. Patent 2043416, 1936.

[97] E. Bjarnason, "Analysis of the filtered-X LMS algorithm", IEEE Trans. Speech Audio Processing, vol. 3, pp. 504-514, Nov, 1995.

[98] E. Bjarnason, "Active noise cancellation using a modified form of the filtered-X LMS algorithm", Proc. 6th European Signal Processing Conf. vol. 2 (Amsterdam: Elsevier, 1992), pp. 1053-1056.

[99] M. Rupp, A. H. Sayed, "Modified FXLMS algorithms with improved convergence performance", Conf. Rec 29th Asilomar Conf. Signals, Syst., Comput., vol. 2, pp. 1255-1259.

[100] M. Rupp, "Saving complexity of modified filtered-X LMS and delayed update LMS algorithms", IEEE Circuits System II: Anal. Digital Signal Processing, vol. 44, pp, 45-48, Jan 1997.

[101] Scott C. Douglas, "Fast Implementations of the Filtered-X and LMS algorithms for multichannel active noise control", IEEE Trans. on Speech and Audio Processing, vol. 7, no. 4, pp. 454-465, July 1999.

[102] Scott C. Douglas, "An efficient implementation of the modified filtered-X LMS algorithm", IEEE Processing Letters, vol. 4, no. 10, pp. 286-288, Oct. 1997.

[103] D.S. Nelson, S.C. Dougles, M. Bodson, "Fast block adaptive algorithms for feedforward active noise control", Pro. National Conference on Noise Control Engineering, (NOISE-CON), University Park, PA, vol. 2, pp. 197-208, June 1997.

[104] M. Rupp, A.H. Sayed, "Robust FXLMS algorithms with improved convergence performance", IEEE Trans. Speech Audio Processing, vol. 6, pp. 78-75, Jan. 1998.

[105] I. Kim, H. Na, K. Kim, Y. Park, "Constraint filtered-X and filtered-U algorithms for the active control of noise in a duct", J. Acoust. Soc. Am., vol. 95, no. 6, pp. 3397-3389, June 1994.

[106] Simon Haykin, "Nonlinear Adaptive Prediction of Nonstationary Signals", IEEE Trans. Signal Processing, Vol. 43, No. 2, February, 1995.

[107] Jens Baltersee, Jonathon A. Chambers, "Nonlinear Adaptive Prediction of Speech With A Pipeline Recurrent Neural Network", IEEE Trans. Signal Processing, Vol. 46, No. 8, August, 1998.

[108] Ashraf A.M.K, Kenji N, "A Cascade Form Predictor of Neural and FIR Filters and Its Minimum Size Estimation Based on Nonlinearity Analysis of Time Series", IEICE Trans. Fundamentals, Vol. E81-A, No. 3, March, 1998.

[109] Ashraf A.M.K, Kenji N, "A Hybrid Nonlinear Predictor: Analysis of Learning Process and Predictability for Noisy Time Series", IEICE Trans. Fundamentals, Vol. E82-A, No. 8, August, 1999.

[110] http://www.astro.oma.be/SIDC/index.html

[111] Seng Kah Phooi, Zhihong Man, H.R. Wu, "Lyapunov stability-based RBF neural networks for nonlinear adaptive predictive coding", 2nd International Conference on Information, Communications and signal processing, ICICS'99, pp. paper no. 421, 1999.

[112] Seng Kah Phooi, Zhihong Man, H.R. Wu, "Nonlinear active noise control using Lyapunov theory and RBF neural network", The 2000 IEEE Workshop on Neural Network for Signal Processing (NNSP'2000), vol 2, pp. 916-925, 2000.

[113] Seng Kah Phooi, Zhihong Man, H.R. Wu, "A New Adaptive IIR Filter with Lyapunov Stability Theory" - The 7th IEEE Singapore International Conference on Communication Systems (ICCS), *CDROM.*

[114] Seng Kah Phooi, Zhihong Man, H.R. Wu, "Designing a fuzzy adaptive NLMS algorithm", International Conference on Artificial Intelligence in Science and Technology (AISAT'2000), pp. 39-43, 2000.

[115] Seng Kah Phooi, Zhihong Man, H.R. Wu, "Complex RBF network with Lyapunov theory training algorithm", International Conference on Artificial Intelligence in Science and Technology (AISAT'2000), pp. 44-49, 2000.

[116] Seng Kah Phooi, Zhihong Man, W. Y. Chui, H.R. Wu, "On-line Identification of nonlinear system using polynomial basis function neural network and Lyapunov

theory", International Conference on Artificial Intelligence in Science and Technology (AISAT'2000), pp. 219-223, 2000.

[117] H. Fan, "New adaptive IIR filtering algorithms", University of Illinois at Urbana-Champaign, PhD Thesis, 1986.

[118] C. Johnson, "Adaptive IIR filtering: Current results and open issues", IEEE Trans. Inform. Theory, 30, no. 2, pp. 237-250, 1984.

[119] H. Fan, M. Nayeri, "On error surfaces of sufficient order adaptive IIR filters: proofs and counterexamples to a unimodality conjecture", IEEE Trans. Acoust. Speech Signal Process., 37, no. 9, pp. 1436, 1989.

[120] K. Steiglitz, L.E. McBride, "A technique for the identification of linear system identification", IEEE Trans. Automat. Control, 26, no. 3, pp. 712-717, 1981.

[121] D. Parikh, N. Ahmed, "On adaptive algorithm for IIR filters", Proc. IEEE, 66, no. 5, pp. 585-588, 1978.

[122] P.L. Feintuch, "An adaptive recursive LMS filter", Proc. IEEE, pp. 1622-1625, 1976.

[123] C. Johnson, M.G. Larimore, "Comments on and additions to an adaptive recursive LMS filter", Proc. IEEE, 65, no. 9, pp. 1399-1402, 1977.

[124] L. Ljung, T. Soderstrom, Theory and practice of recursive identification. The MIT Press, 1987.

[125] S. Mao, P. Lin, "A test of nonlinear autoregressive models", Int. Conf. Acoustics Speech and Signal Proc., 1988.

[126] T. Soderstrom, P. Stocia, "Some properties of the output error method", Automatica, J.IFAC, 18, pp. 93-99, 1982.

[127] T. Soderstrom, P. Stocia, "On some system identification techniques for adaptive filtering", IEEE Trans. Circuits and Systems, 35, no. 4, pp. 457-461, 1988.

[128] H. Fan, "Appliction of Benveniste's convergence results in the study of adaptive IIR filtering algorithms", IEEE Trans. Inform. Theory, 34, no. 4, pp. 692-709, 1988.

[129] B. Friedlander, "System identification techniques for adaptive signal processing", Circuits Systems Signal Process., 1, no. 1, pp. 3-41, 1982.

[130] C . Johnson, "Another use of the Lin-Narendra error model: HARF", IEEE Trans. Automat. Control, 25, no. 5, pp. 985-988, 1980.

[131] M. G, Larimore, J.R. Treichler, C. Johnson, "SHARF: an algorithm for adapting IIR digital filters", IEEE Trans. Acoust. Speech Signal Process., 28, no. 4, pp. 428-440, 1980.

[132] V.M. Popov, Hyperstability of control systems. Springer-Verlag, 1973.

[133] T. Soderstrom, P. Stocia, System Identification. Prentice Hall, 1989.

[134] B. Friedlander, "A modified prefilter for some recursive parameter estimation algorithms", IEEE Trans. Automat. Control, 27, no. 1, pp. 232-235, 1982.

[135] T. Murali, B.V. Rao, "An improved recursive LMS algorithm using a modified gradient filter", Proc. IEEE, 73, no. 8, pp. 1338-1339, 1985.

[136] L. Ljung, "On positive real transfer functions and convergence of some recursive schemes", IEEE Trans. Automat. Control, 22, no. 4, pp. 539-551, 1977.

[137] T.C. Hsia, "A simplified adaptive recursive filter design", Proc. IEEE, 69, no.9, pp. 1153-1155, 1981.

[138] H. Fan, W.K. Jenkins, "New adaptive IIR filter", IEEE Trans. Circuits and Systems, 33, no. 10, pp. 939-947, 1986.

[139] D. Parikh, N. Ahmed, "Sequential regression consideration of an adaptive filtering example", Proc. IEEE, 66, no. 12, pp. 1660-1662, 1978.

[140] S.A. Tretter, Introduction to discrete-time signal processing. New York, Wiley, 1976.

[141] S.A. White, "An adaptive recursive digital filter", Proc. 9th Asilomar Conference Circuits, Systems & Computers, Pacific Grove, CA, Nov. 1975.

[142] http://www.spd.eee.strath.ac.uk/~interact/AF/aftutorial/adfilindex.html